PCT

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : G06F 15/21	**A1**	**(11) International Publication Number:** WO 92/04679 **(43) International Publication Date:** 19 March 1992 (19.03.92)

(71) Applicant: SEER TECHNOLOGIES, INC. [US/US]; Five Penn Plaza, New York, NY 10001 (US).

(72) Inventors: ABBAEI, Manoochehr ; 515 New Brunswick Rd., Somerset, NJ 08873 (US). ANDERSON, Kent, L. ; 2134 N. Military Rd., Arlington, VA 22207 (US). ASH, Rami ; 1416 The Colony, Hartsdale, NY 10530 (US). AVILA, Gregory, Fernando ; 102-40 67th Dr. #2C, Forest Hills, NY 11375 (US). BARTSCH, Paula, L. ; 217 Massachusetts St., Westfield, NJ 07090 (US). BIRDIE, Khurshed, F. ; 165 West End Ave., Apt. 5M, New York, NY 10023 (US). BIRSCHBACH, Michael ; 300 4th St., E. Northport, NY 11731 (US). BLAIR, Mark, H. ; 45 West 60th St., New York, NY 10023 (US). BORROR, Jeffrey ; Five Pond Drive, Waccabuc, NY 10597 (US). BRADLEY, Karen, Susan ; 309 Midland Ave., Metuchen, NJ 08840 (US). BRENNEN, Andrew ; Spectrum Concepts, 150 Broadway, Suite 814, New York, NY 10038 (US). BROWN, Todd ; 36 Cliff Drive, Englewood, NJ 07631 (US). CAMPBELL, James ; 48 Keats Way, Morristown, NJ 07960 (US). CARELLA, Joseph, L. ; 1134 Palmer Avenue, Larchmont, NY 10538 (US). CASE, Stephen, P. ; 80 Candlewood Drive, Murray Hill, NJ 07974 (US). CHIAPPETTA, Wayne ; Brandon Associates, New York, NY 10000 (US). CLAY, Nicholas, John ; 1825 Jackson St., San Francisco, CA 94109 (US). COMMERFOD, JoEllen ; c/o Arthur Andersen, 1345 6th Avenue, New York, NY 10105 (US). CORCORAN, Patricia ; 8898 16th Ave., Brooklyn, NY 11214 (US). CUSWORTH, Richard, A. ; 76 Highview Avenue, Park Ridge, NJ 07656 (US). EISENBERG, Ivy, Mae ; 250 Mercer St. #D1102, New York, NY 10012 (US). FERRUCCI, Charlotte, M. ; 291 Sherman Ave., Jersey City, NJ 07307 (US). FIDUCCIA, Frank, J. ; 3 Glenmere Pl., Old Bridge, NJ 08857 (US). FRIEDMAN, Jacob ; Coopers & Lybrand, 1251 6th Avenue, New York, NY 10020 (US). *(Continued on back of page)*

(54) Title: TRANSACTION PROCESSOR

(57) Abstract

A transaction processor (2) utilized in a multi-environment computer hardware system (1) that permits an integrated way to process and track the many transaction events related to running a business or organization, such as a securities trading firm. The transaction processor (2) permits centralized storage of transaction data for integrated access by programmed modules (6) tracking different transaction events.

(72) Inventors (Continued)

FU, Victor; 94-31 59th Ave., 2J, Elmhurst, NY 11373 (US). GERBER, Mindy; 208 Halsey Ave., Jericho, NY 11753-1602 (US). GESUMARIA, Dennis; 113 Shadowlawn Way, Cranford, NJ 07016 (US). GOLD, Rebecca, Rose; 920 E. 17th Str., Apt. 322, Brooklyn, NY 111230 (US). GOLDFINGER, Irving, M.; 925 Mayfield Rd., Woodmere, NY 11598 (US). GRAHAM, Patricia; 708 Mainsail Lane, Secaucus, NJ 07094 (US). GRAY, Mary; 639-10th St., Brooklyn, NY 11215 (US). GUZMAN, David; 1129 Undercliff Ave., Morris Heights, NY 10453 (US). HALL, Lawrence, Edward; 45-50 160th St., Flushing, NY (US). HAMBURGER, Cynthia, B.; 177 Hillside Ave., Berkeley Heights, NJ (US). HENEY, Nigel, Christopher; Flat 1, 35 Balham Hill, London, SW12 9DX (GB). HOPKINS, Stephen; 5 Oakland Ct. Matawan, NJ 07747 (US). HOROWITZ, David; 2003 E. 22nd St., Brooklyn, NY 11229 (US). HUANG, Zheng-Yi; 605 Thornton Ct., Edgewater, NJ 07020 (US). HUGHES, James; 36 Wilson St., Hartsdale, NY 10530 (US). KAISER, Joseph, Bill; 259-43 149th Rd., Rosedale, NY 11422 (US). KAPLAN, Andrew; 11 Fairhaven Dr. New City, NY 10956 (US). KENNETT, Graham; 175 W. 86th St. New York, NY 10021 (US). KOKIN, David; 5005 Hana Rd., Edison, NJ 08817 (US). KOPF, Richard, A.; 434 E. 75th St., New York, NY 10021 (US). LATTANZIO, Mario; Application Resources, 8 Hampton Way, Woodbury, NY 11797 (US). LEE, Jeffrey, Peng Mun; 345 E. 93rd St., Apt. 9A, New York, NY 10128 (US). LEE, Louisa, Man-Hung; 66 Grandview Place, Montclair, NJ 07043 (US). LEONE, Robert; FD Consulting, 2040 Victory Boulevard, Staten Island, NY 10314 (US). LESHNER, Kenneth, P.; 18 Pittsford Way, Nanuet, NY 10954 (US). LEWIS, Thomas, Jr., K.; 24 Chester Woods Dr., Chester, NJ 07930 (US). LIANG, Adlinna, Y.; 416 White Oak Ridge Rd., Short Hills, NJ 07078 (US). LING, Charlie; FD Consulting, 2040 Victory Boulevard, Staten Island, New York 10314 (US). LIU, Porpeang, Lawrence; 102-12, 65th Ave., Apt. C32, Forest Hills, NY 11375 (US). LUPOWITZ, Mark; Tekmark, 37 E. 29th St., New York, NY 10016 (US). LUXENBERG, Randi, J.; 10 Wooleys Lane 2F, Great Neck, NY 11023 (US). MAFFEI, Eric; 237 E. 79th St., New York, NY 10021 (US). MAIO, Peter, .V.; 25 W. 13th St. 1C-South, New York, NY 10011 (US). MANSKY, Janet, Paula; 225 E., 95th St., 9A, New York, NY 10128 (US). MARKOWITZ, Mindy, R.; 280 Bronxville Rd. Bronxville, NY 10708 (US). McCARTHY, Mark, David; 11 Rutgers Pl., Scarsdale, NY 10583 (US). McINERNEY, Elizabeth, Ann; 732 Hudson St., Hoboken, NJ 07030 (US). MEHTA, Jer, K.; 415 E. 37th St., Apt. 38D, New York, NY 10016 (US). MOSEBACH, Richard, W.; 100 Sixth St., Hicksville, NY 11801 (US). MIERZYKOWSKI, Michael; 27 Cherry St., Malden, MA 02148 (US). NACKASH, Avraham; 215 W. 75th., Apt. 4E, New York, NY 10023 (US). NICHTER, Alvin, Hillel; 524 E. 20th St., New York, NY 10009 (US). NICOLL, David, Charles; 12 Stoney Hill Pl., Linvigston, NJ 07039 (US). NISSIM, Daniel, L.; 20 N. Manor Dr., White Plains, NY 10603 (US). NODAR, Carol; 303 Graham Ave., Brooklyn, NY 11211 (US). NOON, Claire; 30 5th Ave., Apt. 15B, New York, NY 10011 (US). OMANOFF, William; 122 Queens Ct., Massapequa Pk., NY 11762 (US). PAOLUCCI, John, T.; 142 Noel St., Staten Island, NY 10312 (US). PARISI, Anthony; 83 Fulton Ave., Atlantic Beach, NJ 11509 (US). PATEL, Sunilkumar, M.; 575 Buckingham Drive, Piscataway, NJ 08854 (US). POLLARD, William, M.; 5 Countryside Dr., Doylestown, PA 18901 (US). QUINN, Madelyn; 47 Armstrong Ave., Staten Island, NY 10308 (US). RABKIN, Mark; 16 Ivy Ct., Staten Island, NY 1-0309 (US). RAMASWAMY, Nagaswamy; 4 Hickory Ct., Lawrenceville, NJ 08648 (US). RAPPA, Joseph; 24 Jamie Ct., Staten Island, NY 10314 (US). READ, Karen; 511 E. 80th St., New York, NY 10021 (US). REESE, Jonathan, Robert; 51 Elmwood Pl., Short Hills, NJ 07078 (US). RIZZO, S., Anthony; 1675 York Ave., New York, NY 10128 (US). SALWAY, John; 1468 Poulson St., Wantagh, NY 11793 (US). SCHADE, Mary Ann; 325 W. 86th St. 11b, New York, NY 10024 (US). SCHALLUP, Steve; 927 Willow Avenue, Apt. 6, Hoboken, NJ 07030 (US). SCOTSON, Richard, D.; 521 Piermont Ave., Apt. 120, Rivervale, NJ 07675 (US). SHANTZ, Colby; Tekmark, 37 E. 39th St., New York, NY 10016 (US). SHARRY, William, T.; 40 Daly Rd., E. Northport, NY 11731 (US). SHERIDAN, James; 415 W. 47th St. 5RW, New York, NY 10036 (US). SIEGEL, Jay, Lawrence; 80 E. End Ave. No. 17A, New York, NY 10028 (US). SILHI, Kamal; 61-88 Dry Harbor Rd., 5L Middle Village, NY 11379 (US). SIROYA, Basant, K.; 615-11th.St., Brooklyn, NY 11215 (US). SOUTHWORTH, Hamilton, Jr.; 1133 Park Ave., New York, NY 10128 (US). SZETO, Richard; 280 Bronxville Rd. Apt. 2A, Bronxville, NY 10708 (US). TAFET, Hope,. Renee; 11 Riverside Dr., New York, NY 10023 (US). TEASLEY, Carol, Biggers; 352 Herrick Ave., Teaneck, NJ 07666-3153 (US). TEMAN, David; 9 Richland Dr., Springfield, NJ 07081 (US). TERRASI, Daniel; 441 Linden St., W. Hempstead, NY 11552-2514 (US). TIERNAN, Michael, B.; 351 Marine Ave., Brooklyn, NY 11209 (US). TIKTIN, Ross, M.; 125-16 83rd St., Kew Gardens, NY (US). TRIVELLI, Anthony, Christopher; 54 Van Doren Ave., Chatham, NJ 07928 (US). TULCHINSKY, Alexander; 14 Glenn Oaks Ct., Old Bridge, NJ 08857 (US). ULRICH, Joyce, L.; 12 E. 49th St., New York, NY 10017 (US). VANDEWOUDE, John, Edward; 75 Elberta Dr., E. Northport, NY 11731 (US). WALKER, Edward, George; 1425 Harmon Cove Towers, Secaucus, NJ 07094 (US). WEISINGER, Bryan, Alan; 301 E. 22nd St. Apt. 11E, New York, NY 10010 (US). WEISS-MORRIS, Loretta; Tekmark, 37, 39th St., New York, NY 10016 (US). WHEELER, Bruce, Harrison; 95 Mill Spring Lane, Stamford, CT 06903 (US). WIPPER, James, W.; 819 Wesley St., Baldwin, NY 11510 (US). WOJCIEHOWSKI, Ben, M.; 58 Idaho Lane, Aberdeen, NJ 07747 (US). WOO, Doris; 18 Park Circle, White Planes, NY 10603 (US). YOUNG, Yvonne; 616 Longview Rd., So. Orange, NJ 07079 (US). ZANICEK, John; ZIMMER, Leo; The First Boston Corporation, 12 E. 49th St., New York, NY 10017 (US).

TRANSACTION PROCESSOR

Field of Invention

The invention relates to multiprocessor computer
5 systems and, in particular, to a data processing
method and apparatus for integrated gathering and
processing of transactions.

Background of the Invention

Organizations including businesses use computers,
10 associated mass storage systems, and application
software to aid in the management of their affairs.
Records need to be kept, detailing all the
transactions an organization executes. A
transaction may be defined as, e.g., any buying or
15 selling action executed by an organization.
Associated with a transaction are transaction
status events such as sending transaction
information, like an order confirmation, or making
payment for goods received. Transaction records
20 allow organization managers to track inventory and
cash flow. Data stored for each transaction can be
used to maintain an organization's ledger and also
to project future needs.

Securities trading organizations such as investment banking firms and brokerage houses are specific examples of organizations that record data on executed transactions. For those firms,

5 transactions reflect the buying and selling of security products. Transactions are made on behalf of accounts maintained by the trading firm: customer accounts; and the firm's own trading accounts. Customer purchases of securities can be

10 satisfied either directly, using securities from the brokerage organization's own account, or indirectly, with the trading firm acting as a broker. As a broker, the firm will satisfy customer trades by making secondary transactions

15 with other security trading firms. When a security firm makes a trade for its own account or it sells securities from its own account, one refers to the firm as making a principal trade. When the firm acts as a broker, the firm makes an agency trade.

20 Transaction record keeping for security firms presents a challenge, because of the complex details involved with the security products traded and the need for speedy and accurate processing of information on all transactions executed by the

25 trading firm.

Securities trading firms exchange both debt and equity type securities, including currency transactions. The products are extremely varied in their component characteristics (e.g., yield,

30 maturity length, and dividend type). Moreover, the security products traded are constantly evolving. Common security products traded include bonds, corporate stocks, commercial paper, BNMA's, FNMA's, options, futures, high-yield currencies, interest

rate savings, various United States treasury securities and mortgage-backed securities.

When a security trading firm executes a transaction, it needs to record many trade related

5   details, relative to three dates: trade date, settlement date and entitlement date.

On the date that the trade was executed (i.e. trade date) a security trading firm needs to record events of the trade including the type of product,

10   the buying trader, the selling trader, the quantity of the security traded, the price, and the customer or firm account for which the trade was made.

Moreover, some securities require special record tracking. For example, a securities trading

15   organization located in the United States might buy, over-the-counter, the bond of a foreign government, from a security-trading organization located in a third country. For this transaction, payment might be made in the currency native to the

20   trading company from an account located at a designated bank. However, the facial denomination of the bond and the interest generated might be denoted in the currency native to the foreign government. Yields, taxes and any fees owed must

25   also be calculated and recorded. For this example, calculations in three currencies must be maintained for a single transaction.

In addition to trade date details, a securities trading firm also needs to track settlement date

30   transaction details. When a trade is executed, the common practice in the securities industry is not to immediately exchange securities for payment; a

transaction is executed by an agreement to later exchange securities and payment. The time period between the trade date execution of a trade and the settlement date exchange is generally short.

5 Trades of some securities must settle by the end of the business day in which the trade was executed. Trades of other securities can settle in as many as five business days after the trade. For settlement date purposes, security trading firms track

10 information such as the expected date of a trade settlement, the quantity of securities expected or amount of payment due, and the location of the expected security delivery.

Most securities provide entitlements to their

15 holders. Entitlements can be an interest payment on a bond or a dividend declared by a corporation and paid on each share of corporate stock. Entitlements generally accrue on a cyclical basis. A trading firm needs to track entitlement

20 information for all securities in accounts for which the trading firm maintains records and include the entitlement accrual date, the type of entitlement (such as interest payment or stock dividend) in the record for each account.

25 Moreover, there is a special need in the securities trading industry for accurate up-to-the-minute information on all trades executed by a security trading firm. Traders make split-second buying and selling decisions, based on fluctuating market

30 conditions. To execute advantageous trades, the trader must have knowledge of the firm's securities holdings, the competing firms' offering prices and other pricing information.

For example, price information for certain
securities is determined in part by the price of
the firm's most recent sales. The most accurate
pricings are determined by the most recent

5   information on the trades. In addition, some trade
orders can be satisfied directly, using securities
from the firm's own inventory. To make sales from
the firm inventory, traders need to have accurate
and recent information concerning a firm's position

10 in a given security.

Information concerning daily executed transactions
are also used to forecast the end-of-day needs of
the investment firm for cash and securities. For
example, if a firm cannot pay for securities

15 purchased with the funds on hand, it must borrow
funds with an overnight loan. To make accurate
borrowing decisions, treasury personnel need up-
to-the-minute information on trades as they clear.
In addition, if a firm has promised a quantity of

20 securities to be available on a given day and the
firm is short in that position, the firm must
purchase those securities. A firm manager requires
accurate and recent inventory information to make
decisions concerning stock borrowing and loaning.

25 There are special characteristics of the securities
trading industry that make the project of keeping
up-to-the-minute records on executed trades
particularly difficult. For example, the speed
with which market positions change and the number

30 of trades executed in a trading day for each branch
office of a securities trading firm make record
keeping for the securities industry a true
challenge. On a given day, a typical "Wall Street"
trading firm can easily execute tens of thousands

of trades.

In addition, in the present global economy, a
record keeping system needs to be available
twenty-four hours a day. A typical security
5 trading firm trades securities around the clock in
any open securities market such as those in New
York, London and Tokyo. Although each branch
office can perform batch processing at the end of
its business day, all branches of an investment
10 organization together need constant access to on-
line information maintained by a transaction
processing system.

Organizations have employed in the past a number of
computer-based solutions for these transaction
15 processing needs. The currently available
solutions suffer from a number of defects that make
them inefficient and undesirable to organizations
like security trading firms.

The major drawback of current record keeping
20 systems is that they do not process transaction
data in real time -- that is executed trades are
not entered into the system -- as they are made.
Instead, records of executed transactions are saved
until the end of day and entered into the system
25 using batch processes. Batch processing is a
method of scheduling and executing programs where
programs run with no programmer interaction. Input
is fed to the computer in batches. While most
currently available systems are operating in batch
30 mode, they are tied to the processing of the old
trades and no new trades can be added to the
system. Thus, the batching is performed at the end
of each business day ("end-of-day").

However, end-of-day batch processing of executed transactions creates a time lag in updating a firm's data resources, so that the actual firm position in a given product is never accurately

5 reflected until the batch processing is complete. Most currently available systems reflect actual up to the minute information only at the beginning of each new day. But in a business such as security trading, buying and selling decisions based on

10 inventory information must be made during - not at the end of - the trading day. The currently available systems are ineffective in aiding intra-day decisions.

Moreover, the current use of batch processing is a

15 hindrance in businesses where twenty-four-hour, round-the-clock processing capability is required. With batch processing, as it is currently employed, the system is closed to user input during a batch process. For example, a typical securities trading

20 firm has branch offices around the world. Although each branch will close at the end of a business day and each branch can process some transactions and information in batch; the company as a whole remains open all twenty four hours each day, and

25 needs to have its branches share common information. Thus, current systems which will not permit real-time data sharing during batch processing are a hinderance to security firm processing.

30 Beyond the problems of batch processing, transaction processing systems currently available do not provide for full integration of features necessary to process all events related to a transaction. For example, in the securities

trading industry, when an executed trade is entered, there is an expectation of payment either owing or to be received from the trade. The information about the trade can be used for pricing

5 future trades. Further, the information can be used for forecasting -- to determine the need for overnight loans and end-of-day short position security purchases. When payment is finally received or securities are delivered, that

10 information can also be used to reflect the firm inventory holdings or bank account balances.

On currently available systems, those functions of transaction booking, payable or receivable recordings, product pricing, inventory management,

15 and bank account balance maintenance are each performed separately, using distinct non-integrated components. Data stored and used by the one component most often must be re-entered, either by-hand or by-tape, for use by another system

20 component. The current method creates data redundancy and is slow. It cannot provide users with accurate and up-to-the-minute information on the organization.

Inflexibility is a further drawback to the

25 currently available transaction processing systems. In the business of security trading, for example, the products available are constantly evolving and changing. For each security product, the record keeping requirements are different. The currently

30 available systems employ codes that are written into and become part of program statements in the system components. Changing one characteristic of a product requires overhaul of the entire system's software.

Current transaction processing systems fail to
harness the multi-task processing capabilities of
computer technology and the method of cooperatively
distributing software tasks across computer
5 environments. Multi-task processing describes the
simultaneous execution of two or more sequences of
instructions, or one sequence of instructions
operating on two or more sets of data, by a group
of linked computers. Such a computer system will
10 consist of a network of two or more processors.

Multi-task processing machines generally possess
the ability to process events both synchronously
and asynchronously. In engineering terms,
synchronous processing means that, each event, or
15 the performance of each operation, starts as a
result of a clock signal. With asynchronous
processing, each event, or the performance of each
operation, starts as a result of a signal generated
by the completion of the previous event or
20 operation, or by the availability of the processes
required for the next event or operation. In the
context of computer programing, synchronously
scheduled modules are linked such that one module
must wait for the completion of other modules
25 before it can begin again and process the next
task. An example is a program module designed to
wait for confirmation that a subsequent task has
been completed without error. Asynchronously
scheduled modules complete their scheduled tasks
30 and restart without waiting for the completion of
subsequent tasks.

The method of cooperative processing is to break a
computer application into its component functions
and distribute each component to different hardware

environments.

Summary of the Invention

The present invention comprises a transaction
processor, dedicated to recording, maintaining and
5   settling transactions. A transaction can be, e.g.,
any buying or selling event. With basic
information about the transaction stored, the
invention includes the capability to analyze and
forecast using the data.

10   The invention comprises both computer hardware and
computer software elements. Generally, the present
invention comprises a computer system utilized to
process transaction information. The computer
system according to the present invention includes
15   a transaction information entry module comprising:

i.   a trade entry processor having an input to
receive information relevant to a plurality of
        individual transactions and for generation of
        individual transaction records, one
20   corresponding to each one of the plurality of
        individual transactions; and

ii.   a transaction record file coupled to the trade
        entry processor for input and indexed storage
        of each one of the individual transaction
25   records.

Moreover, the computer system according to the
present invention includes transaction record
processing modules coupled to the transaction
record file. Each of the transaction processing
30   modules is arranged to controllably access each one

of the individual transaction records from the
transaction record file. The transaction record
processing modules each include:

i.   a transaction processor for correlating each
5   one of the accessed transaction records to certain
        individual transaction accounts and cumulative
        transaction accounting and inventory records
        maintained by an organization; and

ii.   a preselected set of individual account and
10   cumulative accounting and inventory files
        coupled to the transaction processor for input
        and indexed storage of the correlated
        information to update and record the
        individual transaction accounts and cumulative
15   transaction account and inventory records, as
        a function of the individual transaction
        records.

    Pursuant to a feature of the present invention, a
    communication module is coupled to each of the
20   transaction information entry module and the
    transaction record processing modules to transmit
    notification communications between those modules.
    The notification communications can include
    notification of storage of individual transaction
25   records in the transaction record file. The
    transaction record processing modules are
    responsive to notification communications from the
    transaction information entry module to access the
    individual transaction records from the transaction
30   record file, asynchronously to operation of the
    transaction information entry module, so that entry
    of information relevant to each of the plurality of
    individual transactions and correlating processing

and update and recordation of individual
transaction accounts and cumulative transaction
accounting and inventory records as a function of
the individual transaction records, can be
5 processed in an on-line operation.

A preselected set of reference data bases
containing attribute information relating to data
elements input to the trade entry processor, in
connection with the plurality of transactions, can
10 be coupled to the trade entry processor. The trade
entry processor can access the attribute
information during generation of the individual
transaction records so as to include relevant
attribute information in each one of the
15 transaction records.

One feature of the invention is its use of parallel
and multi-task processing hardware to create a
speedy and totally integrated transaction
processing system. Various time-consuming, but
20 necessary functions of a transaction processor can
be processed asynchronously or in background
without forcing the user to wait for completion.
However, crucial functions can be processed on-line
in real time. The use of asynchronous and
25 synchronous scheduling in multi-task processing
systems gives users the ability to process
transactions on-line and in real time.

Another feature of the invention is the use of
cooperative processing methodology to efficiently
30 parcel the programmed components to hardware
environments directly suited to each component's
function. In a multi-task processing system that
employs different types of computers, such as

personal computers (PCs), and larger computers, the
program elements can be distributed to take
advantage of the particular strengths of each
hardware element. For example, all program modules
5 pertaining to a user interface could be distributed
to the PC environment, while components that
perform many calculations on large amounts of data
could reside on a large mainframe computer.

Because processing can be directed automatically to
10 the processor to which it is best suited, it is not
necessary to select only one processor and data
management system for a system or subsystem. Even
very
small blocks of code can be sent to the processor
15 most suited to the task at hand, whether it be to
provide a user friendly interface, process a high
volume of transactions, or give a user easy access
to data. Large systems can be built to take
advantage of the relatively low cost of PCs and
20 mini-computers. Typical fully loaded costs
(including processor, software, storage,
communications hardware and so forth) per MIP (a
measure of processor power) for a PC, mini-
computer, and large mainframe are $5,000.00,
25 $80,000.00 and $240,000.00 respectively. Not all
computers are capable of solving all computing
tasks. However, using cooperative processing
techniques, processing tasks in the present
invention are distributed to the lowest cost
30 processor capable of handling the job. In addition
to minimizing operating costs, the transaction
processor of the present invention optimizes system
performance by directing tasks to those processors
most capable of providing superior performance.

The use of tightly coupled multiprocessing hardware
and the use of co-operative processing are
important tools not effectively utilized by current
transaction processing systems. The use of multi-
5   task processing techniques can provide speed,
because many processes can be performed at once,
some on-line, some in background. The processes
can also be integrated. The method of cooperative
process distribution allows the components to be
10   efficiently placed in a hardware environment geared
to the component's function.

A third feature of the present invention is the
system of programmed communication elements that
enable the distributed elements of the transaction
15   processing system to exchange data. It is the
communication system that provides the total
integration of features comprising the transaction
processing system.

A fourth feature of the present invention is a
20   collection of centralized databases, containing
general product information accessible to all
component functions of the transaction processor.
Users need only input minimum amounts of
information concerning an executed transaction.
25   The databases provide all the supplemental
information needed by any of the component
functions to fully process the transaction.

A distinct feature of the centralized database
collection is a product classification system. The
30   product classification system of the present
invention provides routines that group products in
many different inverted-tree tables by different
product attributes. This method of tree-table

grouping provides a flexible way of classifying
products for different system use and frees the
user from hard coding those classifications into
the system program modules.

5 The above and other features of the present
invention provides a totally integrated system to
record data on executed transactions. By sharing
data and using a communications system that links
different transaction processing functions together
10 across hardware environments, the present invention
provides a transaction processor that is quick
enough to handle even the needs of a security
trading firm substantially in an on-line direction.

Brief Description of the Drawings and Appendices

15   **I. Drawings**

FIG 1:   depicts a hardware configuration imple-
          menting a transaction processor according
          to the present invention.

FIG 2:   depicts a LAN hardware configuration
20   implementing a transaction processor
          according to the present invention.

FIG 3:   depicts a general overview of the
          transaction processor process flow
          according to the present invention.

25 FIG 4: depicts the general elements of a generic
          distribution mechanism according to the
          present invention.

FIG 4a: depicts a process flow of an object login

for transaction processor elements such as a router or application module.

FIG 4b: depicts a process flow when a user executes an object bind request after login.

FIG 4c: depicts a process flow of an application bind request.

FIG 4d: depicts transmission of data between a PC-workstation and a mini-computer resident application.

FIG 4e: depicts process flow when an application module fails.

FIG 5:   depicts process flow of an application updating files using an IO manager.

FIG 6:   depicts process flow of a store and forward system according to the present invention.

FIG 7:   depicts general reference databases for the transaction processor as implemented for a securities firm.

FIG 8:   depicts a typical product classification tree created by a product classification system module (PCS) according to the present.

FIG 9:   depicts a flow for the creation of the PCS tree of FIG 8.

FIG 10:   depicts a process flow for a trade entry

module according to the present invention, as implemented for a security exchange firm.

FIG 11:   depicts data flow for the trade entry
5   module of FIG 10.

FIG 11a: depicts the process flow for the trade generation module as shown in FIG 11.

FIG 12:   depicts a data process flow of a floor entry module according to the present
10   invention.

FIG 13:   depicts a data process flow for a desk entry module according to the present invention.

FIG 14:   depicts a process flow for breaksheet
15   processing module according to the present invention.

FIG 15:   depicts process flow for a front-end trade inquiry module according to the present invention.

20 FIG 16: illustrates a method of creating integrated payable and receivable records according to the present invention as part of the cash management system module (CAMS).

25 FIG 16a: depicts a flow of control for cash records received by a cash management system module (CAMS) according to the present invention.

activity received from a mini-computer resident module.

FIG 20:   depicts a process flow of a firm inventory module according to the present invention.

5   FIG 20a: illustrates an example of a lot processing file action of the position and balance (P&B) module.

FIG 20b: depicts a lot record movement in handling an "as of" posting to FIFO-based open and
10   closed lots.

FIG 21a: depicts a process flow of the capture transaction function of a customer accounts module according to the present invention.

15   FIG 21b: The process flow of the margin calcualtion, settlement balances, exception processing, notice authorization, debt/credit interest processing, and reporting functions of a
20   customer accounts module according to the present invention.

FIG 22a: depicts a process flow when a trade is booked on a clearance module according to the present invention.

25   FIG 22b: depicts a process flow of the clearance module of FIG. 22a when a trade is paired-off.

FIG 22c: depicts a process flow of the clearance

module when a trade clears.

FIG 22d: depicts a process flow of the clearance module when a failed trade clears.

FIG 23: depicts a high level system flow of a dividends, interest and redemption (DIR) module according to the present invention.

FIG 23a: depicts a process flow for creation of entitlement announcement lists for the DIR module of FIG 23.

FIG 23b: depicts a process flow for a start-up scheduling process according to the present invention.

FIG 23c: depicts a proofsheet entitlement process according to the present invention.

FIG 23d: depicts a proofsheet adjustment process according to the present invention.

FIG 23e: depicts a process flow for a DIR entitlement distribution system according to the present invention.

FIG 23f: depicts a process flow for a DIR receipt and reconciliation system according to the present invention.

FIG 24: illustrates a process flow of a general ledger interface system according to the present invention.

II. Appendicies

Appendix I: object entity relationships for generic distribution mechanism module.

Appendix II: list of possible IO Managerfile access commands.

Appendix III: lists an example of IO manager initial checking routines.

Appendix IV: an example of a data fields from an executed trade file according to the present invention.

Appendix V: an example of data fields used the cash management module of the present invention taken from the executed trade file.

Appendix VI: examples of product position codes for a transaction processor according to the present invention as implemented for a securities firm.

Appendix VII: examples of position codes as they are used in position and balance tables for the transaction processor as implemented for a securities firm.

Appendix VIII: examples of transaction activity and its affect on special memorandum account funds as used by a customer accounts module in the present invention.

Detailed Description

The present invention provides a data processing
system, comprising computer hardware and program
15 elements to process transactions. The present
invention allows continuous on-line processing of
transactions, integrated record keeping and
business function applications.

I. Hardware Elements

20 The present invention can be implemented in any
multi-task processing environment that supports
both asynchronous and synchronous scheduling of
distributed processes. An example of a hardware
configuration suitable for implementing the
25 transaction processor of the present invention is
shown in FIG 1. The figure escribes a "three-
tiered" computer architecture, named for the three
distinct types of processors networked: a mainframe
computer 2, as for example a mainframe sold under
30 the trademark "IBM 3090", a mini or super mini

computer 4, as for example those sold under the
trademark "IBM S/88" or "Stratus" mini-computer,
and a plurality of microcomputer workstations 6, as
for example those sold under the trademark "IBM

5   PC". The PC workstations, which can be located
anywhere in the world, provide data input and local
updating facilities. The PC's also enable a user
to access processes on the larger computers.
Mini-computers, for example those sold under the

10 trademark "Stratus" provide immediate processing
for data input from the PC's.

A Stratus mini-computer environment consists of
multiple Stratus units configured as a single
image, through a networking device, such as those

15 sold under the trademark "Strata-Link" and
"Strata-Net". Fault tolerance is a preferred
feature of the Stratus brand name computer
architecture. Fault tolerance is a construction
technique employing hardware redundancy -- every

20 device such as a chip or a driver has a duplicate
that will take over in the event its counterpart
fails. The mainframe, such as the IBM, houses the
master databases and performs calculations
accessing those large massive storage structures.

25 For purposes of an exemplary embodiment of the
present invention, the processors recommended for
each of those tiers are the IBM mainframe sold
under the trademark "Model 3090", running the IBM
MVS/XA operating system; the Stratus mini-computer

30 sold under the trademark "Model XA200", running the
Stratus Computer, Inc. VOS operating system, and
the IBM micro-computer sold under the trademark
"PS2", executing the Microsoft Corporation
operating system sold under the trademark "MS-DOS."

For further information on these processors or
their operating systems, the reader is referred to
the following publications that are hereby
expressly incorporated by reference: "IBM
5 System/370 Bibliography", document number 6024448;
and "Introduction to VOS", by Stratus Computer,
Inc., document number R00001.

In the hardware configuration above, the PC,
Stratus minis and mainframe are linked by
10 transmission lines. Between PC 6 and Stratus
mini-computer 4 it is recommended that the
transmission lines support the X.25 standard
interface protocol. For links between the
mainframe and the other computers, it is
15 recommended that the transmission lines support the
IBM brand protocol referred to by the name "LU
6.2".

Implementation of the transaction processor of the
present invention is not limited to a three-tiered
20 hardware configuration. The transaction processor
can be implemented using any hardware configuration
that supports parallel and multi-task processing.
It is the parallel and multi-task processing
capability -- where different functions of an
25 application are efficiently distributed -- that
permits the speedy and continuous processing of
transactions as embodied in the present invention.

For example, the transaction processor also could
be implemented using a LAN or local area network.
30 FIG 2 depicts an exemplary implementation of the
transaction processor of present invention in a LAN
environment, such as an Ethernet brand network.
The workstation 10 would access processing

functions, not through a centralized group of
mini-computers, as in FIG 1, but rather to one of
several LAN-minis 12, serving workstation groups.
A workstation group served by one LAN mini-computer

5   could be the users at one district office of a
company, such as a Tokyo branch office. Each
mini-computer performs tasks for its user group 10.
The LAN mini-computer 12 would also facilitate user
links to a mainframe 14 or other additional task

10 processing minis 16. Centralized database
information would be distributed to each LAN mini-
computer to maintain full integration of system
functions.

II. Programmed Elements

15 In addition to the hardware described above, the
present invention comprises a number of program
elements or "modules" and data storage files.

A. Overview

A general overview of the basic process for the
20 present invention is shown in FIG 3. All of the
elements mentioned will be described in more detail
below. The present invention comprises three
different types of software elements: front-end
modules, utility modules and transaction processing
25 modules.

In the PC environment 20 reside front-end input and
user control modules that enable users to run the
present invention. In an exemplary embodiment of
the present invention, it is recommended that
30 interface modules run with an interface program
from the Microsoft Company sold under the trademark

"WINDOWS."

PC resident front-end modules are designed to input
data 22, to view existing data 24 and monitor and
control other transaction processor modules 26.
5 The input modules 26 receive data either on-line
from a user 28, or in batch processes from tape 30
or disk input 32.

Data processing is performed by transaction
processing modules that reside either in a mini-
10 computer 34 or mainframe environment 36. Critical
functions requiring immediate or fault tolerant
processing are performed in the minicomputer
environment 34. Less critical bookkeeping
functions involving larger amounts of data are
15 performed by modules that reside on the mainframe
environment 36.

The present invention also comprises a number of
utility modules to facilitate transaction
processing.

20 A generic distribution module 38 and a PC-based
router module 40 facilitate communication between
PC-resident and mini-computer-based modules. The
generic distribution module 38 serves as a routing
device to channel the flow of information between
25 transaction processor functions resident on the
mini-computer. In addition, the generic
distribution module 38 performs security and log on
functions, and monitors the availability of the
transaction processor's programmed elements.

30 Store and forward modules 41, 42 facilitate
communication between transaction processing

modules that reside in the mainframe environment 36
and those modules that reside either in mini-
computer 34 or PC workstation 20 environments. The
primary store and forward module 41 resides in the

5   mainframe environment 36. A mini-computer resident
store and forward module 42 initiates and controls
the transmission of data from the mini-computer
environment 54. The transaction processing modules
that reside in the mini-computer environment 34

10   such as a trade entry module 48 described below,
feed data to mainframe resident program elements,
using the store and forward modules 41 and 42. All
mini-computer resident feeding systems write
records to a store and forward log file 43 that is

15   read by the mini-computer store and forward module
42. The mainframe resident store and forward
module 41 writes data to a number of massive
storage tables 44. A store and forward notifier
module 46 deposits a message to each mainframe

20   resident transaction processing module that could
be interested in the massive storage table 44
updates.

To facilitate communications from the mainframe
back to the minicomputer, the present invention

25   comprises a call minicomputer module 45.

For purposes of an exemplary embodiment of the
present invention, it is recommended that the IBM
brand relational data base sold under the trademark
"DB2" be used to create the massive storage tables

30   44 that reside in the mainframe environment 36.
For background information on DB2, the reader is
referred to the following IBM publications which
are hereby expressly incorporated by reference:
"IBM DATABASE 2 Introduction to SQL" (document

number GC26-4082); "IBM DATABASE 2 Reference"
(document number SC26-4078); "OS/VS2 TSO Command
Language Reference" (document number GC28-0646);
"TSO extensions Command Language Reference"
5 (document number SC28-1307); "Interactive System
Productivity Facility/Program Development Facility
for MBS: Program Reference" (document number
SC34-2089); "Interactive System Productivity
Facility/Program Development Facility of MVS:
10 Dialog Management Services (document number SC34-
2137) and "DB2 Application Programming Guide for
TSO and Batch Users."

For purposes of an exemplary embodiment of the
invention, it is recommended that Stratus brand
15 operating system VOS files be used to implement the
store and forward log files 43 that exist in the
mini-computer environment 34. The present
invention also includes a number of transaction
processing modules that perform the actual
20 transaction processing and record keeping.
Transaction processing modules that make heavy use
of on-line, real time processing or perform
critical processing that require a fault tolerant
hardware architecture reside on the mini-computer
25 environment 34, such as the fault-tolerant Stratus
brand mini-computer. Transaction processing
modules that access data in the massive storage
tables 44, and do not perform critical real-time
processing reside in the mainframe computer
30 environment 36.

The main input processing module of the present
invention comprises a group of trade entry modules
48. As implemented for a security trading firm,
the trade entry modules 48 receive from workstation

resident input modules 26 all initial data on
trades as they are executed on a trade date. All
trades are initially entered into the transaction
processor through the trade entry modules 48.

5 An executed trade file 50 is the primary storage
area for data concerning executed trades. After
processing data on an executed trade, the trade
entry modules 48 cause an executed trade record to
be written to the executed trade files 50. With
10 the trade record created, the trade is "booked" on
the transaction processor. The trade entry modules
48 notify other transaction processing modules such
as a clearance module 52, described below, of the
existence of a new trade. Other transaction
15 processor modules requiring executed trade data
simply access the central executed trade files 50
upon notification of a newly booked trade.

The trade entry modules 48 perform critical, real-
time trade processing. Thus, for purposes of an
20 exemplary embodiment of the present invention it is
recommended that the trade entry modules reside in
the fault-tolerant mini-computer environment 36.
The mini-computer-resident executed trade files 50
can be created as a group of Stratus operating
25 system VOS files. However, mainframe resident
transaction processing modules also utilize data
stored in the executed trade files 50. For those
modules it is recommended that an executed trade
data table be created in the massive storage tables
30 44. The trade entry modules 48 can write a copy of
each executed trade record to the store and forward
log file 43 for transmission to the DB2 massive
storage tables 44. The store and forward notifier
module 46 alerts other mainframe resident

transaction processing modules, such as a firm
inventory module 66 described below, of the newly
"booked" trade.

Other critical processing functions that require
5 fault tolerant Stratus processing include the
clearance 52 and the cash management systems 54.
The cash management system (CAMS) 54 tracks the
flow of cash for the firm. An integrated payables
and receivables file 56 that CAMS maintains is the
10 central firm cash flow storehouse. The clearance
module 52 records and tracks clearance and
settlement activities for all trades. It receives
input from clearing house banks 58 and updates all
firm positions. In addition, clearance sends all
15 cash receipt records to CAMS 54. A clearance main
file 60 is a central repository of pending
settlement information.

Firm and customer account positions and the
locations of the securities represented by those
20 positions are maintained in position and balance
files 62. Security positions are tracked by
account and location on a trade date and settlement
date basis. A position and balance module (P&B) 64
creates the updates. The major systems feeding
25 data to the P&B module 64 are firm inventory 66,
margins and customer accounts 68, and clearance 52.
The firm inventory module 66 tracks security trades
for firm accounts and calculates profits, losses
and the cost of carry, based upon different lot
30 liquidation strategies. The customer accounts
module 68 maintains customer accounts in a manner
similar to the firm inventory module 66.

A dividends, interest and redemption module (DIR)

70, resides in the mainframe environment 36. DIR
access trade entry module 48 data from the
mainframe resident executed trade table 44 and
monitors all trades for entitlements due. An
5   example of an entitlement is an interest payment on
a bond or a dividend declared on common stock. DIR
70 notifies other programs such as CAMS 54,
clearance 52, firm inventory 66, and margins 68,
sending relevant entitlement information. The
10   entitlement schedules that DIR maintains are the
central location for entitlement information.

Most transaction processor modules described above
send input to a general ledger interface module
(GLI) 72. That module translates all transaction
15   related events into accounting journal entries.
The entries are written to accounting journal files
74 which reside in the mainframe environment 36.

In addition, all of the modules described above
access central reference databases 76, 77 that
20   reside in both the mainframe 36 and mini-computer
34 environments. Those databases contain general
product and other information commonly used by all
program elements of the transaction processor. The
set of central reference databases 76 that exist on
25   the mini-computer environment 34, is an identical
copy of the databases 77 that exist in the
mainframe environment 36.

B. Utility Modules

Important features of the present invention are
30   communication and data distribution modules that
enable the transaction processor to facilitate
communication between modules and to integrate the

processing modules across different hardware
environments.

1. Generic Distribution Mechanism

The present invention includes a generic
5   distribution mechanism module 38 which comprises an
arrangement of cooperative processes that are
responsible for providing the security and
communication links between workstation front-end
modules and program elements that reside in the
10 mini-computer environment 34.

An overall architecture for the generic
distribution mechanism is depicted in FIG 4. The
module comprises several components, each providing
a distinct function. A profiler 80 is the central
15 controlling mechanism of the distribution system.
The profiler 80 views the world of the transaction
processor as consisting solely of objects between
which it provides network links. The definition of
a network object includes hardware elements,
20 program elements, workstation and even users. An
object is any discrete entity that communicates
using the network. The profiler 80 assigns a
unique identification number to each object. The
profilers 80 main function is to enable and manage
25 the functioning of objects that comprise the
communication network. In addition, the profiler
80 of the present invention performs security
access functions.

The actual functions of the profiler 80 are
30 distributed between it and other distinct
programmed objects. However, the profiler 80
oversees and controls the communication-task

objects that are not contained within it. The
profiler 80 manages the other components and the
overall network based on control information and
requests generated by the other objects. Due to
5   the dynamic nature of the network, a great deal of
the control and enabling information is collected
at run-time and combined with static information,
stored in a database 82, described below.

All objects communicate with the profiler 80 using
10 messages called network primitives. Primitives
comprise a given object's data transmission
prefaced by a request header.

The profiler 80 processes these primatives
which include, for example:
15   1) object-login requests;
             2) object-logout requests;
             3) object-bind requests;
             4) application-bind requests;
             5) object-failure-notification; and
20   6) date-and-time requests.
Object-login requests are generated by network
objects when they attempt to log into the system.
As described below, the profiler 80 assigns to the
object an identification number and adds that
25 object into the network, using the information
provided by the object's request primitive.

An object-logout request signifies that the
requestor is attempting an orderly network sign-
off. If valid, the profiler 80 will logout all
30 objects dependent upon the requestor object. The
profiler 80 then de-registers the object from the
network as will appear.

As described more fully below, an object-bind
request signifies that a network object is
attempting to establish a link with another object.
If the request is valid, the profiler 80 will
5   return to the requestor, the object identifier for
the remote object requested and its availability
status. In addition, the profiler 80 will update a
list that specifies the dependencies between
objects.

10   An application-bind request is similar to an
object-bind request. The difference is that only
users can make application-bind requests. The
profiler 80 responds to an application-bind request
by returning the names, object identifiers, and
15   availability status of all applications that a
particular user has access to through a particular
workstation. The list of applications to which a
user has access to at a particularly workstation is
determined by a security access files in a database
20   82, described below.

Object-failure notification is a signal to the
profiler that some object has failed to respond.
On receiving notice of the reported failure, the
profiler 80 notifies each object logically
25   dependent on the failed object. The profiler 80
will then update the network database described
below to reflect the object failure. The procedure
as described below is similar to that of an object
logout.

30   A time-of-day request is used by many different
program elements when they send data. The profiler
80 returns the time-of-day.

All of the requests described above are initiated
by the network objects. The profiler 80 is a
real-time, event-driven process that is continually
executing, while awaiting to receive network
5 requests. Upon receiving a request, the profiler
80 performs rudimentary validation, such as
checking that the primitive contents are consistent
and then forwards the data to specific request
processing routines.

10 Associated with the profiler 80 is a relational
database 82 and a group of standard query language
procedures 84 that the profiler uses to access the
information stored there. All static control
information needed by the profiler 80 to create and
15 maintain the communication network is kept in the
database 82.

The database information can be classified into two
categories: network and security related
information. Logically, the network information is
20 organized by individual network objects as follows:
        Object Network Name.
        -Object ID:
        -Object Type:
            > USER
25   > WORKSTATION
            > FEATURE APPLICATION (e.g. CAMS)
            > INFRASTRUCTURE COMPONENT (e.g.
    Profiler).
        -Logical name
30   -password (only for object type: USER)
    In the database, network object data is grouped by
    relationships. For example, user-object
    information is stored according to user group and
    workstation group relationships. The user group

relationship consists of

    -user members

    -application members

    -workstation group members.

5 The workstation group relationship consists of:

    -workstation members

    -application members

The objection relationships are fully depicted in
Appendix I. For more information on relationship-
10 type modeling, the reader is referred to co-pending
U.S. patent application Serial No. 444,060, filed
November 30, 1989 and entitled, "Computer-Aided
Software Engineering Facility," which is hereby
expressly incorporated by reference.

15 The database is initiated and maintained by a local
administration facility 86. This module gives
security administration personnel direct access to
the database. Users can add objects or
relationships and change or delete existing ones.

20 While the profiler maintains the network, it is the
other component: routers 88, 90, 92, 94, PC
interface modules 96, queues 98, 100, 102, 104,
106, 108, 110, 112, X.25-call-routers 114; and PC-
based routers 116 that actually move the
25 information throughout the system.

The PC-workstations and mini-computers are linked
by standard transmission cable lines 118. For
purposes of an exemplary embodiment of the present
invention, it is recommended that lines capable of
30 supporting the X.25 protocol standard be employed.
X.25 is an industry term specifying the interface
between data terminal equipment and data circuit
equipment.

Located in each PC workstation environment is a PC
router module 116 that facilitates communications
from PC to minicomputer environment. All
applications that reside in the workstation

5 computer environment invoke the PC router 116 to
receive and transmit data. For each workstation
there is a copy of the PC-router that is configured
to a specific X.25 line.

On a mini-computer, as for example a Stratus brand

10 mini, X.25 transmission lines are connected to
hardware ports, X.25 gateways 120. For each port
in the present invention two program modules are
attached: a PC interface (PCI) module 96 and an
X.25-call-router module 114. An X.25 call-router

15 114 is a process used to establish a virtual
circuit beween the PCI and the X.25 gateway. A
virtual circuit is an independent logical path
between two end-entities created for the purpose of
exchanging data. Each X.25-call router 114 is

20 assigned to send messages to a specific PCI module
96.

PCI's 96 are protocol gateway processes that
provide the communication link between the low-
level X.25 gateways and high-level router

25 communication system. Protocol is an industry term
defining the conventions or rules for the format
and timing of messages sent and received. PCI's
alleviate any format or timing differences of a
message as it was sent from the PC and as it should

30 be received in the Stratus environment. In
addition, PCI's 96 perform routine security
checking for all message primitives received by the
X.25 gateways 120.

The central communication component in the present
invention is the router system (routers 88, 90, 92
and 94). Routers are high speed switches. A
router's primary function is to receive message
5   requests from network objects and send them to
specific destinations. A router is programmed to
report any switching failure to the profiler 80.
In addition, routers 88, 90, 92 and 94 maintain
statistics used by the profiler 80 on message
10   traffic throughout the system.

PC-based applications 122, mini-computer-based
applications 124 and 126 and the profiler 80 all
communicate through the router system. PC-based
applications 122 can only access the router system
15   through an X.25 gateway 120 and a PCI protocol
module 96. The router system comprises the group
of router switches 88, 90, 92 and 94 to which the
profiler 80 has supplied access locations for all
the objects in the system. Each router 88, 90, 92
20   and 94 is assigned by the profiler 80 to service
particular network objects, such as PCI modules 96
or applications e.g. 124 or 126. Those assigned
objects are local to the designated router. A
local object can directly send and receive messages
25   to and from another local object using only their
common local router. For objects that are not
local, an object must forward messages from its
local router to the object's local router. The
profiler 80 provides all connectivity information
30   to create the router system and can dynamically
change router service assignments, depending on the
use statistics that the routers provide. That
function known as "warm starting," provides for a
restart of the router system.

Messages are sent to, from and between routers
using a series of queues 98, 100, 102, 104, 106,
108, 110 and 112. On the mini-computer, queues are
data storage areas, managed by the core of the

5 operating system, allocated from the core storage
pool, and used for communications between
processes. For those mini-computer-resident
network objects: routers 88, 90, 92 and 94,
profiler 80, applications 124 and 126 and PCI's 96

10 -- there is an input queue dedicated to each
specific object. To send data, the sending object
writes the data to the receiving object's queue.
For example, an application 126 does not send data
to its router 94 directly, it sends data to the

15 router's queue 108. To receive data an object,
like a router 94, reads its queue.

The process flow of the generic distribution
mechanism functions such as an object login, a
user-login, an application bind, an object failure

20 and a normal data transmission are depicted in FIGS
4a-e.

The data flow depicted in FIG 4a is representative
of the login for application modules, PCI's, and
routers. An application module 124 generates a

25 network login request and forwards it to the queue
104 of its default assigned router 90. The router
90 reads the message header and determines that it
is a profiler action request. The router 90
forwards the request to the profiler's queue 100.

30 Profiler 80 reads the message of its input queue
100 and then queries the database 82 for
information concerning the application. The
profiler checks the information received from the

application against the security information in the
database 82. Once validated, the profiler 80
directs all routers (88, 90, etc.) to establish
connectivity routes to the application from each of
5   their domains. This is accomplished when the
profiler 80 sends to each router (88, 90, etc.) a
message primative containing information that the
routers will use to update their connectivity
mapping tables. Each router module 88 has within
10   it a connectivity table, storing within it the
address of modules local to each other. The
profiler 80 sends each router message to the input
queue 102 of its local router 88. The local router
68 updates its own table, using the message
15   addressed to it, and forwards the other messages to
the other routers.

Once connectivity to the application is
established, the profiler 80 sends to the
application a login acknowledgement. The message
20   contains an object-identification number that is
now assigned to the application 124 and which the
application will now use in future requests. Thus,
the object login function provides the important
function of network hook-up and security clearance
25   for the transaction processor. The login process
is similar for all objects.

Immediately after a network object has successfully
logged into the network, it must dynamically bind
any remote application with which it needs to
30   interact. This is accomplished by sending object-
bind requests to the profiler 80. The binding
process for both user-objects and application-
objects is similar. When logging in, a user can
bind for use only those applications that are

available both for his or her use and for display
at a given workstation. A security manager has
established security clearances for both the
workstation and the user. Those access clearances
5   are stored in the relational database 82. For all
applications, the database stores a list of all
other applications it uses.

FIG 4b depicts the process flow when a user
executes an application bind request after login.
10 A PC-resident router 116 takes the request and
attempts to establish a virtual circuit with the
PCI 96 dedicated to that particular workstation,
using the X.25 gateway 120 and the PCI's X.25-call
router 114. The virtual circuit establishment
15 process is described more fully below.

Once the PCI 96 receives the full request
transmission, it writes the data to the queue 102
of its local router 88. The router 88 sends the
request to the queue 100 of the profiler 80.

20 The profiler 80 takes the request from its queue
100 and then accesses the database 82 to determine
which applications this particular user and
workstation combination can have access to.
Associated with the user-object data is a list of
25 all applications that the user has security
clearance to access. Related to the workstation-
object information in the database 82 is a list of
application functions that are authorized for use
on the particular workstation. The profiler module
30 80 matches the two relationships to determine what
applications the user can access from the given
workstation.

-42-

Using a message primitive, the profiler module 80
updates a security access list that is used by the
workstation's PCI 96. The profiler 80 will next
generate a response that contains the logical names
5   of all processes available to the user. The
response is forwarded through the router system to
the PCI 96 and back to the workstation 118.

The case of one application binding itself to
another is described by FIG 4c. The requesting
10 application 124 sends an object-bind request to the
profiler 80, using the router system and queues
104, 90, 102, 88, and 100. Associated with the
application-object data in the database 82, is a
list of other processes upon which the application
15 object 124 depends. The profiler 80 checks the
status of each of the needed applications, and adds
the requesting object (i.e. application 124) to the
dependency lists maintained in the database 82 for
each of the objects it will use. If the binding is
20 successful the profiler 80 will send an
acknowledgement to the requesting application 124,
using the router system.

Once an object is registered to the network and
successfully bound, it may send, transmit and
25 receive messages to communicate with other program
modules.

The transmission of data between a PC-workstation
application and a mini-computer based application
is depicted in FIG 4d. A PC-based application 122,
30 such as the front end of the trade-entry system of
FIG 3, gathers data from user input to send to the
mini-computer based customer trade-entry module
referred to in FIG 4d by reference numeral 126.

The application sends the data to the PC-based
router 116, which will create a message primitive
and attempt to send the data to the mini-computer-
based PCI 96 across a virtual circuit.

5   The PC router 116 initiates the creation of the
virtual circuit by generating X.25-protocol-
requests and sending them to the mini-computer.
The signal is accepted by the X.25-call-router 114
which works to establish the virtual circuit
10   between the PC-based router 116 and the PCI 96,
dedicated to that workstation. When the PCI 96 is
free for processing, the X.25-call-router 114 sends
the request to the PCI 96 and a circuit is
established.

15   As implemented on a mini-computer such as a Stratus
brand computer, the PCI 96 employs a VOS supplied
API function to read the message transmission
through the X.25 gateway 120. Once the API
function is initiated the X.25-call-router 114 is
20   completely by-passed until the circuit fails and
needs to be re-established. Re-establishment is
completed in a way similar to the initial
establishment. The API mentioned above is
described in standard VOS documentation.

25   When a PCI 96 receives a message, it acts only as a
protocol gateway -- it simply forwards the message
primitive to the queue of its local router 102.
The router 88 receives the message and reads the
header to determine the message's destination.
30   Aware that the application is not a module to which
it has direct access, the router 88 will forward
the message to the queue 108 of the local router 94
of application B 126. Application B's router 94

determines that application B 126 is a local object
and it forwards the message to application B's
queue 110.

Application modules also use the communication
5   network to exchange data, as FIG 4d again shows.
In passing data from application A 124 to
application B 126, as for example the trade entry
system notification to CAMS of a newly booked
trade, application A 124 generates a message,
10   placing the destination address of B 126 into the
header.   The message is transmitted to the queue
104 of A's local router 90. Application A's router
90 determines that application B 126 is not local,
and forwards the message to the queue 108 of B's
15   local router 94.   B's local router 94 reads the
message and forwards it to application B's queue
110.   Application B 126 will subsequently process
the data and return a response.

In the event of an object failure --
20   application workstation, or even a PCI -- the
router that noticed the failure reports it to the
profiler 80 which will clean up the failure and
notify related objects.   This procedure includes
deleting all connections to the object and
25   preparing for the object's re-initialization.   FIG
4e also depicts the process flow when an
application module fails.   The same scenario
applies in other fa__lure instances such as when a
user fails to sign-off or a utility object such as
30   a PCI fails.

In FIG 4e, application B 126 attempts to access
failed application A 124.   A's router 90 with the
request from application B, sends a message to

determine whether application A 124 is not running
or if its queue 112 is full. Both are failure
conditions. When application A 124 does not accept
the request, A's router 90 sends a failure-

5 notification to the queue 100 of the profiler
module 80.

The profiler 80 validates the message using the
database 82 and performs a number of other steps.
First, it instructs all routers (88, 90, 92, 94,

10 etc.) to delete connectivity to the failed object.
The profiler 80 accomplishes the task by sending to
each router (88, 90, 92, 94, etc.) new connectivity
table inserts that omit links to the failed object.
The profiler 80 will next log-out all objects that

15 are dependent upon the failed object. The profiler
80 performs the same steps as above, for each of
the objects appearing on the failed object's
dependency list. As described above, dependency
lists for all network objects are created by the

20 profiler 80 and stored in the database 82.

In the event that a workstation or communication
link to a workstation 118 fails, the PCIs 96 tied
to that workstation would report the failure to the
profiler 80 in the same way the router did in the

25 previous example, with the same outcome. In the
case of a failed workstation, the profiler 80 will
log-out all applications dedicated only to that
user.

2. IO Manager

30 Distinct from the generic distribution mechanism
module 38 described above, a feature of the present
invention is a uniform method of file access used

by all applications resident on the mini-computer, as depicted in FIG 5. To access data in a file, according to the present invention, each transaction processing module 130 calls upon a copy

5 of a utility module referred to as IO manager routine 32 to access files 136. One copy of the IO Manager 132 is bound to each transaction processing module 130. Each copy of the IO Manager 132 has associated with it one or more files comprising an

10 IO control table 134. The IO control table 134 contains access codes and data relevant to all the files that a particular transaction processing module will access. The IO control table 134 allows a generic copy of the IO manager module 132

15 to perform as an access module dedicated to a specific transaction processing module

IO manager module 132 has a set of commands -- read, write, insert (update), start, commit and add that correspond to file access commands provided by

20 the operating system. A list of sample file access commands (for a system implemented in the Stratus) is shown in Appendix II. It is the IO manager 132, rather than the transaction processor modules 130, that handle the different file access constraints

25 imposed by a given operating system. As implemented in the three-tiered hardware environment depicted in FIG 1, the IO manager 132 is an external procedure module coded in a high-level language such as PL/1.

30 Associated with the IO manager module 132 for each application is the IO control table 134. Those tables 134 contain location addresses for files accessed by a transaction processing module 130 plus information concerning the organization of

those files. Any restrictions on file access imposed on transaction processing module 130 access by system management, are also contained in the IO control table 134, as well as a flag on whether

5 store and forward logging is to be performed in addition to accessing a particular file. The concept of store and forward logging is explained more fully below.

The process flow of an application updating a

10 record in a file using a copy of the IO manager module is shown by FIG 5. An transaction processing module 130 such as the trade entry system in FIG 3, passes to its copy of the IO manager 132, an instruction to complete the file

15 access, (e.g., the file name, the information to be updated, and the task command (i.e. update)). The IO manager module 132 will reference its IO control table 134 to find the location of a desired file 136 and other file update information such as the

20 restrictions imposed upon access to that file by system management.

Using the IO control table 134 information, the IO manager 132 will pass all access commands on to an appropriate operating system file access routine

25 138 for processing. These operating system access routines 138 perform the actual file access for data input and for output (IO). However, before the call is passed to those sub-routines, the IO manager module 132 performs some initial input

30 validation. Appendix III shows the extent of this initial checking. Once the check is complete and valid the operating system performs the corresponding file IO 138.

The IO manager module 132 creates an audit trail by first reading the file 136 record and writing a copy of both the current record and the updated version to an audit log file 140. The IO manager

5   module 132 next updates the accessed file 136, and if file indexes exist, the IO manager module 132 will write to a file index log 144. An asynchronous indexing module 146 reads the file and updates the modules.

10   If required, the IO manager module 132 will also write the data to the store and forward log file 142 as illustrated in FIG 3. As will be described more fully below, the store and forward modules transfer data from the minicomputer to the

15   mainframe environment. The IO manager 132 updates the store and forward log 142 only when it receives a specific instruction to do so from its controlling transaction processing module 130.

The IO manager module as embodied in the present

20   invention presents a number of distinct advantages. Because IO manager modules 132 perform the actual file access, using operating system commands, the applications programmer is freed from the need to be well-versed in the intricacies of file access

25   for a particular operating system. Moreover, with only one IO manager module, an IO manager call is the same no matter what routine uses the IO manager or what file access request is required.

The IO manager's 132 use of an IO control table 134

30   permits security parameters to be easily placed on file access. With the use of tables, the security parameters can be easily changed, too. In addition, the audit log function guarantees that

there is a complete record of all file changes.
Finally, the IO management system of the present
invention permits flexibility for file design
change. Should the need arise to use a different
5   file system, the change can be implemented quickly
-- by changing only the code for the IO manager
master copy and possibly the IO control tables for
the transaction processing modules. The code of
each transacting processing module program need not
10   be changed.

3. Store and Forward

A third distinct communication utility module of
the transaction processor according to the present
invention is store and forward. The basic function
15   of store and forward is to take data from a
globally accessed file on a mini-computer system 34
and transfer the data across a communication link
to a database and program modules that reside on
the mainframe 36. With the present invention
20   implemented in the three-tiered hardware
environment depicted in FIG 1, PC's 6 provide data
input and local updating facilities and enable the
user to access the larger computers for inquiries;
mini-computers 4 provide immediate results for
25   larger groups of data; and the mainframe 2 houses
massive storage tables and program modules that
perform calculations with large amounts of data.
It is the store and forward modules 41, 42 of the
present invention that handle the movement of data
30   from mini to mainframe environments.

The basic information unit exchanged between
hardware environments under the present invention
is a bundle. Store and forward does not transfer

separate records. A bundle consists of the update
activity corresponding to one logical unit of work
from a workstation command or batch process.

The process flow of the store and forward modules
5   is shown in FIG 6. Mini-computer resident
transaction processing modules 150 and workstation
resident front-end modules 152 can reference
mainframe transaction processing modules using
store and forward. A transaction processing module
10   150 using its IO manager 154 stores information in
the mini-computer-resident store and forward log
file 43. In the present invention, the store and
forward log file 43 exists on a separate Stratus
brand mini computer and is accessible to all
15   transaction processing modules using the mini-
computer-linking hardware 158 such as Stratus
Strata-Link hardware.

Asynchronously, a mini-computer-resident store and
forward sorting module 160 sorts through the store
20   and forward log file 43 to gather bundles of
information to send to mainframe ports 170. The
sorting module 160 sorts transactions into
different priority classes and stores them on
separate logs 162, 164, 166 by priority.

25   A mini-computer-resident communication module 168
next sends a bundles of logged transactions to the
mainframe. Using an IBM brand mainframe as an
example, the data is sent to CISC operating system
region of the mainframe.

30   Store and forward maintains a number of
transmission lines using a bisyncronous protocol
format such as an IBM Logical Unit 6.2 Protocol or

an bisyncronous IBM 3780 Protocol.

For each transmission line attached to the
mainframe ports 170, a program module 172 monitors
the transmission function. That program module 172
5   gathers complete messages from the mini-computer
168 workstation or PC-resident 152 modules. The
monitoring modules 172 will write messages to
communication ASCII format files 174 and notify the
mini-computer or workstation resident communication
10   module 168 of the message receipt.

During the transmission of information bundles, the
mini-computer or workstation resident communication
modules 168, 192 communicate with the mainframe-
based modules 172. The mini-computer resident
15   store and forward module computer 168 or
workstation resident 152 module initiates the
procedure; the mainframe module 172 controls it.
For example, the mainframe module 172 instructs the
mini-computer module 168 where in the log to re-
20   start, in gathering a full message bundle and which
steps to repeat.

Asynchronously, a mainframe resident store and
forward controller module 176 reads the ASCII code
file and invokes one or more translation modules
25   178 to translate the data from one file format to
another, if necessary. For example, in a
transmission of data from a Stratus brand mini-
computer environment to an IBM mainframe computer
environment, the data must be translated from ASCII
30   character format to the EBCIDIC format that is used
for file storage in IBM brand computers.
Translated records are stored in an EBCIDIC format
log file 184.

-52-

A group of files 180 known as "key files" are used
in the handling of data on the mainframe. All
static translation information resides in the key
files. In addition, on-line updated information

5   such as the date and time is maintained there,
because every bundle that arrives from the mini-
computer is time stamped. The key files also store
any re-transmission information on a given bundle.
The line monitoring modules 172 uses key file

10 tables 180 to track the transmission of the
information bundles between hardware environments.

To make the translation, store and forward
translation modules 178 access a number of key
files 180. A key file contains information,

15 concerning the field structure for every mini-
computer-based file. The translation modules
utilize the key file information to map the mini-
computer file bundles and locate the bundle fields
that require translation. The translator modules

20 178 also use key file information to decipher
specific fields. For example, in translating
between ASCII and EBCDIC formats, a translator
module 172 determines that the first five bytes of
character contains information and must be

25 translated, while the next four bytes represent a
four word, binary number that needs no translation.
The translator monitor modules then store all
translated bundles in an EBCIDIC format log file
184.

30 The Store and forward modules completes their
function by sending the data to massive database
tables 186 maintained on the mainframe. Reference
tables located in the key files 180 contain
information to map the EBCIDIC file 184 record

information for the transaction to its appropriate
place in the massive storage tables 44. An updater
module 188 accesses the EBCIDIC file data 184 and
"points" the information, using the key file
5   mapping information, into the massive storage files
44. The actual "pointing" is performing by
"pointing" modules 190 built to access a specific
table in the massive storage area 44. Good data is
available to any object with mainframe database
10 access such as mainframe resident transaction
processing modules 192 or users making database
inquiries from workstations. Store and forward
also employs a notification module 46 to alert
other mainframe transaction processing modules 192
15 of the massive storage table 44 updates. Those
transaction processing modules 192 will access the
massive storage table 44 to obtain the information.

To perform the data translation between file
formats such as ASCII and EBCIDIC, a user must
20 specify, beforehand, the file structure for the
massive storage table and the mini-computer
resident file between which data bundles will be
transferred. For purposes of an exemplary
embodiment of the present invention, it is
25 recommended that a one to one or one to many file
relationship exist between the mini-computer files
and the mainframe computer massive storage 44
tables.

C.   Reference Databases

30   **1.   Database Descriptions**

The present invention includes reference databases
76, 77 that comprise central files storing general

-54-

information used by transaction processing modules.
For example, an implementation of the present
invention for an investment institution might have
the following general reference databases as shown
5   in FIG 7; a floor broker master database 200; a
producer master database 202; a trading accounts
master database 204; a product master reference
database 206; a customer accounts reference
database 208; a firm price management database 210
10   and a figuration formula database 212. Using the
example of the present invention, implemented in a
three-tiered hardware environment, two exact copies
of these databases are maintained, one in the
mini-computer environment and one in the mainframe
15   environment. (See FIG 3 at 76 and 77).

A unique database feature of the present invention
is a group of product classification trees 214 that
are created for use by different transaction
processing modules from a product classification
20   front-end module 216 that accesses data in the
product master database 206. Other front-end
modules 218 allows users to access database
information directly.

The floor broker master database 200 keeps
25   information on registered exchange floor brokers
who trade securities for a firm. That database 200
enables a security trading firm to maintain such
general information about the brokerage firms and
their traders such as the firm's name, address and
30   tax identification number. The floor broker master
database 200 maintains flexible broker fee rate
tables that apply to an exchange generally or to a
specific broker or firm. Flat rates that apply to
individual brokers or brokerage firms are also ·

maintained. The floor broker master database 200
contains data to calculate discounts that
specialist brokers sometimes offer.

The producer master database 202 is a storehouse of
5   data concerning all registered exchange producers
who execute trades for the trading firm -
salespeople, traders - anyone who must be
registered to deal in securities through
organizations such as NASD but who is not a floor
10 broker. The producer master database 202 keeps
data on producer sales pool affiliation,
registration, and regulatory exam pass information.

The trading accounts master database 204 stores
data on an investment firm's trading and bank
15 accounts. Trading departments and desks, as well
as individual traders are associated in the
database with the firm's trading accounts.
Reference tables are maintained in the database for
trading account validation.

20 The product reference master database 206 maintains
reference information associated with all the
products purchased or sold by an investment firm.
Domestic and international security products that
comprise this file include but are not limited to
25 equities, debt, options, futures, mortgage-backed,
securities synthetics, and money markets. For
foreign securities, international settlement
calendars as well as country and currency of issue
information is maintained for each product.

30 Associated with the product master database 206 is
the product classification system (PCS) and the
collection of product classification trees 214 that

PCS module 216 creates for general access by
transaction processor modules 220.

The product classification system provides the
ability to install, modify and determine product
5 classification schemes for each security product
listed in the product master database.

The product classification system (PCS) is a method
of categorizing products according to product
attributes. As embodied by the present invention,
10 a PCS tree file 214 comprises a number of inverted
tree hierarchy-tables and commonly accessible
subroutines to traverse those trees and access the
data in them.

The product master database 206, described above,
15 contains different attributes for each product
listed. For example, a security product, such as a
bond, could have these attributes:
* U.S. (country of origin);
* coupon interest;
20 * base currency - U.S. Dollar;
* price; and
* Federal Government Security.
The various transaction processing modules of the
present system need to classify the product
25 differently according to its different attributes.
For one system, the need might be to distinguish
between foreign and United States country of origin
securities:

Foreign Securities:	United States
30	Securities	
* Stock X (Canada);	* Stock A (EXXON);

-57-

       * Gov't Bond (West Germany);   * Gov't Bond (Fed.
                                         U.S.);

       * Municipal (Hamburg);   * Municipal
(N.Y.C.);

5   Another system may require a classification of
    products that distinguishes between governments and
    municipals and other securities:

         Government Securities: Other:
         * Gov't Bonds (Fed. U.S.)   *Stock A (EXXON)
10   * U.S. Municipal (N.Y.)   *Stock B
    (foreign)
         * For. Municipal (Munich)
         * For. Bond (West Germany)

    A product classification tree 214, grouping
15 products by country of original is shown in FIG 8.
    A process flow for the program modules that
    comprise PCS is depicted in FIG 9. Referring to
    FIG 9, an inverted tree construction module 300
    allows users to specify the various formula
20 relationships to build a tree via a workstation
    resident front-end module 308. These user
    specified rules are shown in FIG 8 at 252-276; they
    are the basic nodes to construct the tree.
    Identification codes for references to the product
25 master database are the leaf nodes of the tree.
    (See 280-294). Traversing modules, FIG 9, at 302,
    permit a transaction processing module 304, such as
    the trade entry module described in FIG 3, to
    traverse a PCS tree and obtain the information on
30 the classified products. With the product numbers
    found, the transaction processing modules access
    the product master database 306 for complete
    information on each product.

Many transaction processing modules of the present
invention, such as the trade entry module and the
firm inventory module, described in FIG 3 use one
or more PCS trees to process data. Many
5 transaction processing modules share PCS trees for
common operations. All products in the product
master database 206 are listed in every PCS tree.
Where only certain product classifications are
relevant, a not applicable, "N/A" field at the
10 beginning of a node marks irrelevant branches, as
shown in FIG 8 at 296.

With the product classification system as used in
the present invention, there is no need to hard
code product attribute classifications into the
15 codes of transaction processing modules as is done
currently.

The product classification system also supports
on-line changes to the PCS trees. FIG 9 shows the
process flow for an add, delete, or change to the
20 product attributes.

Through the front-end, on-line module 308, a user
can modify the product master database 306 --
either adding or deleting a product or changing a
product attribute. Product master maintenance
25 module 310 receives the input via the generic
distribution mechanism module 38 and invokes a
notification module 312 to notify the PCS module
314, and every feature system that uses PCS 304 of
the database change. The generic distribution
30 mechanism module 34 as described above (See FIG 4),
performs the notification on the mini-computer, and
on the mainframe, the store and forward module
notification (See FIG 6) performs notification..

-59-

The PCS module receives notification and invokes
either an add 316, delete 318, or modify 320 module
to traverse every existing PCS tree and update the
tree. Transaction processing modules 304, such as
5 DIR 70 are responsible for reclassifying all
transaction data that would be affected by a
product classification change, as will be described
below. The transaction processing modules 304
utilize the new PCS trees to make the changes.

10 The product classification system of the present
system presents a flexible and efficient method for
on-line product classification alterations. PCS
eliminates the need to rely on product "type" codes
often used in the past to classify securities.
15 Those systems had a distinct disadvantage in that
the product codes were necessarily written into the
program's logic. When a new security product is
added in those systems, or a characteristic is
changed in one of these already existing products,
20 the program code for the application had to be
changed. With the system of the present invention,
product codes are not written into the applications
code. Changes required to implement a new product
for altered business conditions are therefore easy
25 to implement and do not require extensive
reprogramming.

Referring again to FIG 7, the customer accounts
reference database 208, contains reference data
associated with a firm's customer accounts.
30 Information is maintained to accommodate numerous
delivery instructions, multiple salesperson
coverage, security transfer instructions, trading
authorizations and compliance papers. Data are
maintained to denote whether an account is a parent

account or a sub-account of a particular parent.
Examples of the categories of information include
names and addresses, standing delivery
instructions, institutional delivery information,
5   telex confirms, financial profiles, trading
authorizations, futures and options information,
legal documentation information, credit limits, and.
broker database cross-references.

The firm price management database, 210, provides
10 an integrated, globally accessible repository of
all pricing information used by the transaction
processing modules. The database also provides
information for real-time collection and
distribution of prices. Pricing information is
15 kept in the currency denomination native to the
security products being sold. The database stores
the currency exchange information necessary for
conversion. The database also contains a
collection of security prices and currency exchange
20 rates gathered from external data sources. Outside
sources include Pricing services such as "J.J.
Kerry", "EOM" "IOSI" and "CSJ" "EOM", "IOSI", "JJK"
and "CSI". Internal pricing sources include input
from the trade entry modules, and manual on-line
25 price inputs. Price entries in the firm price
database 210 are associated with entries in the
product master database 202.

The figuration database 212 is a collection of
subroutines to make common calculations. As the
30 present invention is implemented for a securities
trading firm, the figuration database 212 can
include routines to calculate yields for fixed
income, equity, future and option securities.

-61-

As implemented in the three-tiered environment of
workstation 6, minicomputer 4 and, mainframe 6,
(see FIG 1), the databases described above will
exist in a centralized repository in the mainframe
5   77, and in addition, a duplicate copy 76 will be
stored in the minicomputer environment (See FIG 3).
The mainframe is the central repository for all
transaction data used by the system. However,
because transactions are processed by the quicker
10 processing, fault-tolerant minicomputer, a working
copy of all databases described above are kept in
the mini-environment.

For purposes of an exemplary embodiment of the
present invention, it is recommended that the IBM
15 relational database sold under the trademark "DB2"
be employed for the database on the mainframe. For
purposes of an example embodiment of the invention,
it is recommended that one or more Stratus brand
fault tolerant mini-computers be employed to
20 perform fault tolerant processing. On that mini-
computer it is recommended that stratus operating
system VOS files with indexed files be used to
implement the database.

2. Front-End Database Related Features

25 As will be described below, the all programmed
feature modules of the present invention access the
general databases. However, the present invention
also contains programmed modules to give a user
direct on-line access and allow the user to create,
30 update and view the databases, and, in addition,
perform calculations on the database information.

The product master front end module (FIG 9, 308)

-62-

also provides a user inquiry facility. The product
master reference database (FIG 9, 306) is
referenced by presenting to the user a series of
questions about a product he or she wishes to
5   locate. The user is queried about the
characteristics he or she seeks in the searched for
security product. For example, the system can find
the product a user requires by asking for its
symbol, strike price, or expiration month. To
10 access product master information, the product
master front-end module (FIG 9, 308) invokes the
product master traversing module (FIG 9, 302).

The product master front-end module (FIG 9, 308)
provides a help facility to assist the user in
15 searching the database for a security where a key
is not known. The module 308 constructs a search
key into the product master database 306 by type of
product and returns those securities that best fit
the supplied description.

20   3.   Firm Price Management System

The firm price management system of the present
invention provides the user with a front-end
feature module (FIG 9, 218) to access the firm
price management database (FIG 9, 210). As
25 described above, the firm price management database
is the depository of all pricing information used
by the firm's computer applications. To access the
firm price management database 210 the front-end
module 218 comprises: an external and automatic
30 price capture module 220, a trader price-marking
module 222 and a yield/commission calculator 224.

The price capture module 220 maintains and updates

the security price and currency exchange database
files from a variety of external data sources.
Collection of prices is a two-step process. Prices
are first gathered intra-day and at the end of the
5   market day. Then the information is sorted and
specific prices are selected for storage. Internal
sources of prices from executed trades are also
used to update the firm's price management database
210.

10 The trader price-marking module 222 related to the
firm price management database 210 allows the user
to manually enter a price for a security. The
trader price-marking module 222 allows a trader to
establish and enter a price for a selected
15 security. The trade price marking module 222
permits a user to select a method of pricing, for
example a treasury code or pricing by utilities to
determine the price of a security.

The present invention also includes
20 yield/commission front-end modules 224 that permit
traders to use the figuration database routines 212
to calculate security yields using current price
database information. The yield/commission
calculator module 224 aids traders in arriving at
25 pricing estimates before trades are executed. The
modules can be used as a replacement to stand alone
devices such as the "Monroe Bond Yield Calculator".
By utilizing a group of centrally accessed
databases, the present invention integrates the
30 pricing function into a single transaction
processing system. In addition firm traders can
make pricing estimates with greater accuracy,
because the pricings are computed using actual firm
prices that are updated on-line.

-64-

D. Transaction Processing Modules

As depicted in FIG 3, the transaction processing
modules of the present invention are programmed
elements that perform the actual transaction data
5   processing. Each transaction processing module is
linked to the user front-end modules and other
transaction processing modules through the generic
distribution mechanism modules and the store and
forward modules, as described above. File access
10   for each feature module in the mini-computer
environment is performed by a dedicated IO manager
as described above. The communications network
permits the integration of the functions and
sharing of data between transaction processing
15   modules.

1. Trade Entry Transaction Processing Modules

The prime task of the transaction processor is to
track transactions entered into by an organization.
A cornerstone function of the present invention is
20   performed by the modules that receive input on
newly executed trades. In the example
implementation of the present invention for a
securities trading organization, transactions are
executed at customer account desks, at interfirm
25   trading desks and, on market exchange floors. An
exemplary embodiment of the transaction processor
for a securities trading firm would incorporate the
trade entry modules 48 as depicted in FIG 10.

The trade entry module 330, described below, will
30   be the central entry point for all executed
transactions. The executed trade files 50 created
by the trade entry 330 module contains records that

represent the official "booking" of trades. The
executed trade file 50 is also the master source of
trade-related information for all subsequent
transaction processing modules, such as the cash
5 management system module 54 (CAMS) and the
clearance module 52, as depicted in FIG 3. As the
data fields from an example executed trade file in
Appendix IV show, a great amount of data concerning
a trade is stored in the executed trade files.

10 However, to supplement the accurate building of the
central executed trade file, the present invention
can include two additional transaction processing
modules to capture trade information directly from
floor trading and desk trading operations. Data
15 collected by both a floor entry module 334 and a
desk entry module 336 is matched against the
executed trade file data 50 by a breaksheet
processing module 338. The breaksheet processing
module 338 processing will enable trade entry
20 operators to quickly find and correct the
inaccuracies of executed trade file.

i. Trade Entry Module

The trade entry feature module is the central entry
point for all executed customer or firm security
25 trades.

The method of the present invention for entering a
trade is illustrated in FIG 11. Data concerning an
executed trade is entered by a trade-entry operator
340. In the three-tiered environment, as depicted
30 in FIG 1, entry operators enter data on a
workstation 6. A workstation based trade entry
front-end module 342 accepts the data from the

-66-

keyboard of the workstation and a type checking
module 344 performs field and data-type checking.
If the data is acceptable, the generic distribution
mechanism 38, as described above, is invoked by the
5   workstation based front-end module, 348. The
interactive PC-based router module 40, a pier to
pier communication routine described above, sends
the data to the mini-computer-resident generic
distribution mechanism 38, also described above.
10   That router system sends the data to the trade-
entry system queue 350.

The trade entry wake-up module 352 reads the data
from the queue 350 and sends it to a validation
module 354 for error-checking. The validation
15   module 354 accesses data from the general reference
databases 356 to validate the trade. For example
the validation module 354 would check the product
description against data from the product master
database to determine if the product actually
20   exists. The routine would also check trader data
to determine if the trader was authorized to make
the trade.

Trade net monies are next calculated for each
trade. A figuration module 358 calculates --
25   principal, interest, discounts, commissions. For
example, these calculated figures are summed to
figure the net monies for the trade. The system
can calculate trade net monies in four different
currencies: according to base, notice,
30   consolidated, and settlement difference. Currency
conversion tables located in the general reference
database 356 are used to make the calculations.

A trade entry derivation module 360 next performs

data derivations and population, patching
information gaps with derived information and
filling the data fields. For example a trade entry
operator might enter "M" for the account of
5   customer: Merrill Lynch. The derivation module 360
would use information from database such as the
product master 202 to completely build the customer
data field information, supplying "Merrill Lynch"
and other relevant information concerning that
10   account.

The general database information used by the trade
entry systems is generally gathered by the system
and stored in one or more trade entry extract files
357. The trade entry system database extract files
15   are created in a trade extract builder batch
process 359 that takes all general reference
database file data 76 relevant to trade entry
modules and writes it to indexed files. The
extract files 357 provide speedier access to common
20   data than would be referencing the larger general
databases.

As described above, notification of updates to the
general reference are sent to transaction
processing modules (See FIG 9). The trade entry
25   extract builder 359 receives a notification of
general reference database updates. The extractor
builder module 359 then accesses the general
reference databases 76, reading the specific
updated fields that are relevant to the trade entry
30   transaction processing modules and updating the
general reference extract files 357. The extract
builder module 359 processes in background. The
trade entry processing modules such as the
validation 354, figuration 358 and derivation 360

modules receive all general reference data through
the general reference extract files 357. The
extract files are built to speed reference database
access.

5   With the trade data validated, derived and figured,
a unique trade transaction number is assigned, and
in certain instances trades are split by quantity
into one or more trades of smaller quantity by a
process known as trade generation. A trade
10 generator module 362 handles the process of
breaking a larger transaction into smaller trades.
For example, trades that expected to settle using a
device known as "The Federal Wire Service" can only
clear in denominations up to a certain dollar
15 amount. Trades for securities having a dollar
amount greater than the Federal Wire Service limit
must be split into several trades of smaller
denomination. The process flow for trade
generation in the trade generation and trade number
20 assignment module 362 is described more fully
below.

Once the trade has been validated, figured,
derived, and generated, it is "booked." The
booking process entails writing the collected
25 information into an intermediate log file 366, and
returning a favorable acknowledgement message to
the user. A trade booking module 364 performs the
update to the intermediate log file 366, and a
notification module 367 returns the acknowledgement
30 message. Asynchronously, an updater module 368
reads the records kept in the intermediate log file
366 and copies the data into the executed trade
files 50 and the store and forward log file 43.
The executed trade file 50 is a central storage

repository used by all mini computer-resident
transaction processing modules needing data on
newly booked trades. Associated with the executed
trade files 50, are one or more index files 374

5   that permit quick access to all executed trade
information. An index manager routine 376,
initiated by the updater module 368, performs the
index file 374 updates. A batch index log 375
contains the relative record number of the executed

10   trade file record to index. The store and forward
log file 43 is a storage area for data to be
shipped to feature modules residing on the
mainframe environment. An IO manager module 378,
as described above in FIG 5, executes all file

15   access. Appendix IV lists the data fields for
exemplary executed trade files for a security
trading firm.

The final step of trade entry is to notify other
transaction processing modules of the newly booked

20   trade. This task is performed using a notifier
module 380 which creates messages to send to other
modules via the generic distribution mechanism
module 38.

The trade generator's process flow is depicted in

25   FIG 11a. The trade generator module of FIG 11 at
362 is represented by two submodules. A
determination module 380 makes an initial
determination into trades of smaller quantity. To
make this determination, the determination module

30   380 accesses product and settlement information
from the general reference extract files 356. If
the trade must split a trade generator module 384
is called. The generator module 384 loops, copying
the data concerning the trade into smaller-sized

-70-

trades. Each smaller trade will be "booked" as
before using the book trade module 364 that is
described in FIG 11. Each small trade will receive
a unique table identification number. The split
5   trades are linked by identification number.

ii. Floor Entry Module

To satisfy customer trade obligations, a securities
trading firm can purchase securities either in
markets for regularly traded securities or in
10   over-the-counter markets from dealers. The floor
entry modules of the present invention receives
input on all executed floor trades. The data
reflects trades that have been executed on behalf
of a securities trading firm in regulated markets
15   such as the New York Stock Exchange. The data
compiled by the floor entry module 334 is used to
reconcile against data from the executed trade
files.
The process flow for the floor entry modules is
20   depicted in FIG 12. Data is gathered by a
workstation-resident 6, front-end module 390,
either online from user input 392, in batch from
taped records of programmed trades 394, or other
stored trade records. The generic distribution
25   mechanism 38, described above, receives the data in
the minicomputer environment 4 and routes it to the
mini-computer queue 400 dedicated to the floor
entry system.

From this point the floor entry module would
30   process in a manner similar to the trade entry
system described above. (See FIG 11). Trades
records will be validated 402, figured 404, and
populated 406, using data extracts file 408 created

from the general reference databases 76. The trade
is next written to floor entry intermediate file
412 by a floor entry booking module 414. An
updater module 416 using its IO manager module 418,

5 will write the information to a floor entry trade
file 420 and the store and forward log 43. That
data will be reconciled against executed trade data
during the breaksheet processing phase to be
described below.

10   iii. **Desk Entry Module**

In addition to securities sold in general markets
like the New York Stock Exchange, securities are
also bought and sold in specialized over-the-
counter markets. Trades executed on behalf of a

15 security trading firm in over-the-counter markets
are input into the transaction processing through
the desk entry module 336 and the trade entry
module 330. The process flow for entering a trade
using the desk entry modules is depicted in FIG 13.

20 The desk entry modules in a manner similar to the
trade entry system and the floor entry system as
described in FIGS 11 and 12. Referring to Fig 13,
a desk entry front-end module 430 receives trade
data from a desk entry operator. The PC resident 6

25 module 430 sends the data to the minicomputer 4
environment for processing. The generic
distribution mechanism 38 facilitates the
communication and places the information in a mini
computer environment memory queue 434 dedicated to

30 a desk entry 438 wake-up module. The trade is then
checked by a validation mode 436. A validation
module 436, a derivative module 439 and a
figuration module 440 all work to build a complete

desk-entry file record. Those record-building
procedures are performed by accessing data extract
files 442 created by an extract builder module 443
from the general reference databases 76. The trade
5   is then written by a booking module 446 to a desk-
trade log-file 448. In background, an updater
module 450 writes the records from the intermediate
log file 448 into indexed desk entry files 452. In
addition, the updater module 450 writes the record
10   to the store and forward log file 43. File access
is performed using an IO manager 456 module.
Entries in the desk entry files like those in the
floor entry files will be used to reconcile against
entries in the executed trade files.

15   iv. Breaksheet Processing Module

The function of the breaksheet processing module
338 is to match the records of the floor entry
files and the desk entry files against entries in
the executed trade files to find discrepancies in
20   the executed trade files. Trade records are
maintained when trades are executed either by floor
traders or desk traders, the current practice is to
write executed trade information on slips of paper
and pass those slips to trade entry operators who
25   will input the slip information on to some trade
entry system. Problems with that current system
include the fact that the slips are lost, or never
created. Sometimes inaccurate information is
written on the slips. The volume of trade
30   execution and the speed at which trades are
executed create the inaccuracies. However, there
is no guarantee that the trades as input into the
trade entry system are accurate.

The present invention alleviates the risk
discrepancies by utilizing output of the floor
entry module 334 and desk entry module 330 in
conjunction with the output of the trade entry
5 module 330. The processes of reconciliation of the
floor entry 420 and desk entry 452 files with the
executed trade files 50 is known as breaksheet
processing.

FIG 14 depicts the process flow for breaksheet
10 processing. In a background process a
reconciliation gathering module 460 selects
transaction records from the executed trade files
50 and potentially matching records from either the
floor entry files 420 or the desk entry files 452.
15 The reconciliation gathering module 460 performs
file access using its copy of the IO manager module
468.

Potential matches are sent to a trade matching
routine 470 that performs validation tests to
20 insure that the gathered trades match. For example
the matching routine will examine all trade
generated split trades to match them against a
single desk or floor file entry. In matching, the
match trade module 470 accesses general product and
25 broker data from the general reference databases
76. Unmatched trades from the executed trade file
50 and trades from the desk and floor entry files
420 and 452 that do not match against any executed
trade file entry are listed in an unmatched trade
30 file 474. A reporting module 476 creates a
breaksheet report 478 for researching the unmatched
trades from entries in the unmatched trade file
474.

-74-

v. Front-End Trade Inquiry Module

The present invention provides the user with a
front-end capability to view trades as they have
been booked.

5   The process flow for that trade inquiry function is
described in FIG 15. With a front end module 480
resident on the workstation environment 6, a user
can request data on a single trade or multiple
trades. The generic distribution mechanism module
10 38, described above, routes the information request
to the mini-computer based 4 inquiry modules.

A retrieval trade information module 484 first
attempts to retrieve data on the requested trades.
Using an IO manager module 486, the retrieve module
15 484 will search the executed trade files resident
on minicomputer 50 and the mainframe database 490
for trade information. If trade information
exists, it is presented to the user using a display
window tailored to the specific type of security
20 492. A display module 493 sends information to the
workstation to create the displays 492.

In addition to basic trade retrieval, a user has
other options. A trade management review module
494, displays trades ba.ed on user input search
25 criteria and updates trade approval and time stamps
for a given trade. A multiple trade search module
496 displays further trades based on the user input
search criteria. In addition, the retrieve module
484 will display: all cancel and correct history
30 for a single trade 498; all "give-up" history for a
trade 500; all "when issue" trades related to a
trade 502; and, for internally numbered trades, all

trades bearing the same trade sequence as the
displayed trade 504. An option control module 506
loops presenting options 506 until the user exits.

2. Cash Management System Module

5 The cash management system module (CAMS) 54
depicted in FIG 3 comprises an integrated series of
programmed element modules designed to provide a
central, controlled environment for processing and
recording cash receipts and disbursements. The
10 CAMS modules perform timely reconciliation of bank
cash flows with a securities trading firm's
transaction records. The CAMS modules also provide
to firm treasury officials current cash balance
information and timely cash projections.

15 Implemented on a three-tiered hardware environment,
the modules comprising the CAMS module of the
present invention would reside in the mini-computer
hardware environment 4.

CAMS module has three basic functions relating to
20 the flow of cash within an organization. First,
the CAMS module must record payables and
receivables that are expected from business
transactions. An example of this function occurs
in the securities trading industry when a trade is
25 booked - the expectation of a payable or a
receivable must also be recorded. A second
function is to record the receipt or payment of
cash and reconcile that cash flow against the
payable and receivable files. The third function
30 is to provide users with an up-to-the minute record
of the current cash balances in existing accounts
and to show projected cash totals based on the

-76-

expected payable and receivables. Each of these
functions is described in more detail below.

i. **Payable and Receivable Management**

CAMS receives data on payable and receivables from
5   any system where there is the expectation of cash
flow, such as from the trade entry modules 48 or
the dividends interest and redeemable module (DIR)
depicted in FIG 3. The method of creating payable
and receivable records to mark expection of cash
10 flow is shown in FIG 16.

A cash management system interface module 510
receives notification that the trade entry module
330 has just booked a trade and CAMS must generate
a payable or receivable record. Notification of
15 the creation of an executed trade file is sent from
the trade entry module 330 to the CAMS interface
module mini-computer queue 514, using the generic
distribution mechanism 38, depicted in FIG 3. The
CAMS interface module 510 receives the relative
20 record number of the newly booked trade as it is
written in the trade entry executed trade files 50.
The CAMS interface module 510 calls a trade entry
file accessor module 520 to retrieve information
from the relevant executed trade file entry 50 that
25 is pertinent to cash management, such as the amount
of payment due or the name of the bank account that
is to receive payment. The file accessor 520 is a
module dedicated to locating data in the executed
trade file 50. The fields in the executed trade
30 file 50 used by the CAMS module are shown in
Appendix V. The accessor module 520 retrieves the
data and passes it to the CAMS interface module
510, which in turn, passes the information to a

CAMS log writing module 522. The CAMS log writing
module 522 sends the data to a CAMS intermediate
log file 524 and notifies initial CAMS processing
module 526 by sending the relative log file record

5 524 to a queue 528 accessable by the initial CAMS
processing module.

The initial CAMS processing module 526
asynchronously wakes to read the CAMS log file 524.
The wake-up is triggered by an event such as the

10 passing of time or the input of a number of new
records to the initial CAMS processing module's
queue 528. In implementing the present invention
on a Stratus-brand mini-computer, the present
invention would employ the Stratus-brand "wait

15 event" function available as part of the Stratus
operating system.

To determine where to begin retrieving records in
the intermediate log file 524, the wake-up module
526 first retrieves the relative record number of

20 the last record read in the log intermediate file
524, from a maintenance file 530. The initial CAMS
processing module 526 next passes relative record
numbers for all unprocessed log records to a CAMS
data retrieving module 532.

25 The data retrieving module 532 retrieves the actual
data from the CAMS log file 524 and determines
which processor will handle the data. Many
different systems send data to the CAMS
intermediate log file 524. For each type of data,

30 CAMS employs a different processing module to
create payable and receivables.

A module for creating payable and receivable

records for a newly processed trade is shown in FIG
16 at 534. The payable and receivable creation
module 534 first assembles the data and then writes
it, via a CAMS IO manager module 536, to an
5   integrated payable and receivable file (IPR) 56.
The IPR file 56 contains a record for all payable
and receivables expected throughout the system. A
copy of the record is also placed on the store and
forward log 43. The store and forward
10   communication module, depicted in FIG 6, takes the
data and sends it to mainframe environment
transaction processing modules that also require
CAMS module data.

With the data placed in the integrated payable and
15   receivable file 56, the payable and receivable
creation module 534 updates other files that are
used for the bank totals and ledger keeping
functions of CAMS.

First, the payable and receivable creation module
20   534 initiates an update to an integrated payable
and receivable total files 540. The IPR total file
540 is used in making bank total projections, the
third function of the cash management module. The
totals are stored in different index files. To
25   make the updates, the CAMS payable and receivable
creation module 534 accesses information in one or
more product classification trees 214, using a
product classification traversing modules 216. The
payable and receivable creation module 534 sends
30   the information to be written by the CAMS IO
manager 536 to a log file 548. Totals are also
kept for each bank location. Bank location
information is stored in the trading account master
database 204. Asynchronously, an IPR total manager

module 552 takes the data from the log file 548 and
using information stored in the trading account
master database 204, updates a bank total file 540
and an index file 554, utilizing the CAMS IO
5   manager 536.

Indexes are also maintained for the CAMS IPR file
56 under various keys. To maintain processing
speed the index files are updated in background.
The payable and receivable creation module 534
10 writes index information to a log file 558. An IPR
item index manager 560 writes the indexing
information to an index file 562. The IPR item
index manager 560 uses the CAMS IO manager 536 and
an asynchronous index updating module 566 to
15 complete the task. To determine what was the last
index updated from the log file 558, the IPR item
index manager 560 maintains in a maintenance file
570 the relative record number of the last record
taken from the log file 558. The CAMS IO manager
20 536 accesses the maintenance log 570 for the IPR
item index manager 560.

Finally, the creation of integrated payables and
receivables creates a need to update the general
ledger interface module depicted above in FIG 3.
25 The payable and receivable creation module 534
sends transaction data to the general ledger
interface module 72, that will be described more
fully below.

The payable and receivable creation module 534
30 acknowledges successful processing of the booked
trade data by updating the CAMS intermediate log
524 and a "successfully processed" file 574.
Notification is then sent to the trade entry module

330 on the CAMS processing success.

The CAMS modules comprise a number of checks to
insure that data is processed accurately. When the
cash management modules are initially notified of
5   incoming payable and receivables, the CAMS initial
processing module 510 first reads the CAMS
intermediate log file 524 to see if the executed
trade record to be processed has previously been
added. Even if the new record has already been
10   added, the CAMS initial processing module 510 will
still call the execute trade file accessor module
520 to add the record to the CAMS intermediate log
file 524. However, the record will be marked to
show that it is a duplicate copy of the existing
15   record. This method is used to protect the
integrity of the system and insure that only one
record is processed.

Also, before an actual processing module such as
the payable and receivable module 534 is invoked,
20   the CAMS retrieving module 532 first calls a
checking module 576 to parce through the
"successfully processed record" file 574 and check
that the record to be processed has not been
processed previously. If that record or its double
25   has been processed, the data retrieving module 532
will not invoke any of the CAMS processing modules,
such as the payable and receivable creation module
534 and processing will terminate.

ii. Recording Cash Flow

30   The second function of the CAMS is to record the
flow of cash from the bank accounts and to
reconcile that cash flow against the IPR file 56.

Data for the flow of cash is input to the system
from a number of source applications. Data can
also be input manually. This method is used when
the firm receives a physical check. However, cash

5   is mostly transferred through interbank wire
transfers. CAMS receives wire transfer information
from tie in services with banks. The wire
information is received directly or it is received
indirectly through updates from other transaction

10   processing modules such as the clearance module 52
as depicted in FIG 3.

The CAMS cash flow process is described in FIG 16a.
Much of the initial processing of the cash flow is
exactly the same as the processing of a payable or

15   receivable record modules as depicted in FIG 3.
Notification that money for a trade has exchanged
is received by the log writing module 522. A
workstation front end module 582 accepts data on
physical checks accessed or sent. The clearance

20   module 52 sends data on wire transfers of cash.
Input is sent to the CAMS log writing module 522 by
the generic distribution mechanism 38. The CAMS
log writing module 522 writes the information into
the CAMS intermediate log file 524. The CAMS

25   initial processing module 526 is awakened at the
happening of an event or the passage of time. The
module 526 sends the relative record numbers of the
new records to the CAMS data retrieving module 532.
Asynchronously, the CAMS data retrieving module 532

30   reads the data from the log file 524 and invokes
the CAMS checking module 576 which determines if
the record has already been processed. The CAMS
checking module 576 reads the "successfully
processed record" file 574 to perform the

35   validation. The CAMS initial processing module 532

determines that the data concerns a cash flow and
sends the data to a cash clearance module 580.

The cash clearance module 580 performs a number of
processing steps. First, the cash clearance module
5   580 matches the cash movement data against a
corresponding record from the integrated payable
and receivable file 56. The payable or receivable
file record will be updated with the cash flow
information. The record in the IPR file 56 will
10 reflect the sum still outstanding.

In addition to updating the IPR file 56, the cash
clearance module 580 also updates a number of other
files. Record of the movement is placed in the
store and forward log 43 to be moved via the store
15 and forward module 41, 42 to the transaction
processing modules residing on the mainframe.
Second, the cash movement is posted to a cash
activity file 582 that provides a record of all
movement of cash within between accounts controlled
20 by an organization.

Third, the cash clearance module 580 creates a
cross-reference for each record in the cash
activity file 582, matching it's record against one
or more file record that detail the nature of the
25 transaction. The relative record number from the
cash activity file 582 is matched with a
corresponding relative record number from the IPR
file 56. The cross-reference is kept in a CAMS
cross-reference file 584. Details for all cash
30 activity file records 582 can be located using this
cross-reference method. The file access of the
cash clearance module 580 is initiated by the CAMS
IO manager 536.

When the receipt of cash update comes to CAMS from
another transaction processing module, such as the
clearance module 52 depicted in FIG 3, the details
of the cash receipt have already been stored by the
5   feeding system. In these cases, CAMS will not
create a record in the cash activity file 582.
Instead cash clearance module 580 expects to match
a provided relative record number to a detail file
of the feeding transaction processing module (e.g.
10   clearance 52) with the relative record number of
the IPR update placed in the cross-reference file
584.

However, as described above, cash movement can be
entered manually, through a front-end input module
15   582, where an operator keying into CAMS inputs the
details of a received check. In that case, cash
movement is posted to the cash activity file 582 in
the same manner as if the cash movement was
reported by the clearance module 52. But, there
20   will be no system provided detail file record to
cross-reference with the posted cash activity file
record. In cases of manual input, the cash
clearance module 580 will create a CAMS activity
file 582 record which can be cross-referenced
25   against the CAMS IPR file 56 record.

Once the main file updating activities are
complete, the cash clearance module 580 next
initiates updates to the bank totals files and the
general ledger. In addition the module performs
30   cross-index updating for the cash activity and
integrated payable and receivable files.

First the cash clearance module 580 accesses
product classification system files 214, utilizing

-84-

the product classification system modules 216.
This information is used for indexing updates. In
addition, the cash clearance module 580 accesses
information in the bank location file 204.

5   The cash clearance module 580 initiates updates to
the payable/receivable totals files 540, by writing
the information into a log file 548. The
asychronous processing IPR total manager 552 reads
the data from the log and updates the IPR totals
10   file 540.

The cash clearance module 580 updates information
on the bank account balances, writing information
to a log file 588. Asynchronously, a location
account balance manager module 590 reads the log
15   file 580 and writes the information to a location
balance file 592, using the CAMS IO manager 536.
To track the update progress, the location balance
manager module 590 records the relative record
number of all log file records accessed in a log
20   file 596.

The cash clearance manager 580 next updates the
keyed record file indexes for both the integrated
payable and receivable file 56 and the cash
activity file 582. The information is written to
25   an intermediate log file 591. The intermediate log
file 591 is asychronously read by a cash item index
manager module 595. The cash item index manager
595 obtains the location for the next index updates
in an index file maintenance log 597. That cash
30   item index manager 595 then passes the relative
record number of the update and the index update
data to an index update module 598. The index
module 598 writes the IPR index file record in the

-85-

actual index file 596 using via IO manager 597 and
returns to 595 the relative record number of the
next free index file record in the file 596. The
cash item index manager 595 places that update in
5 the maintenance file 597. File access for the
indexing procedure described is performed utilizing
the CAMS IO manager module 536.

Finally, as described above in FIG 11, CAMS will
post the cash movement data to the general ledger
10 module 572. And as before, the success of the
transaction processing is maintained in the cash
management system log 524 and the "successfully
processed" file 574.

iii. Account Totals and Cash Projections

15 The third function of the cash management system is
to display current bank account totals and to
project future cash totals for the end-of-day and
next-day. Those functions of the CAMS system allow
fund managers to make funding decisions with
20 greater accuracy.

The bank position and forecasting system is
illustrated by FIG 16b. An officer authorized to
handle fund management can access the data in the
IPR total files 540, the IPR item index file 562,
25 and the bank account totals file 592. Those files
provide the user with the current bank account
totals. A cash position and forecasting module 595
accesses those files, using the CAMS file manager
modules 552, 560, 590 and the CAMS IO manager 536.
30 The next day and end of day projections are
computed by examining the CAMS IPR file 56 for
receipts that will become due. In addition, the

-86-

cash activity file 582 and the cash activity
cross-reference files 584, provide factual details
concerning the payable and receivables records in
the IPR files 538.

5   3.   Position and Balance Module

The position and balance (P&B) module 64, and the
P&B files 62 of the present invention provide a
centralized, integrated database for all firm
position data. Using the example of a security
10   trading organization, position data relates to all
securities owned by the customers or firms that are .
maintained by the firm at various security
depository locations. Position and balance files
also maintain information on the status of the
15   positions -- for example indicating which
securities are available for the firm's use in
collateralizing a loan. The combination of the
entity executing the transaction, product involved
in the transaction, account on whose behalf the
20   transaction was executed and position of the
product comprise the definition of a security
position. Assume for example that a customer
purchased and paid for 100,000 shares of IBM common
stock, through a firm's New York office, where the
25   stock is currently held at its vault for
safekeeping. Position and balance files record
that customer's position in IBM as well as the
status and location of the securities by entity
within the firm.

30   Position codes are indicators of the type of
position maintained by a given record in a position
and balance table. Appendix VI contains a listing
of position code examples for the present invention

as implemented for a securities trading
organization. Position codes can be used for
different purposes depending on their use in the
P&B tables as shown by the chart in Appendix VII.

5   For example in the P&B location files 606, 612
position codes such as "BOX", "FREE", "MXRP" and
"SFK" indicate actual physical locations of
security positions while "F/R" (filed to receive),
"F/D" (failed to deliver") indicate an actual

10 position in the securities, but do not reveal the
physical whereabouts of the securities. Those
position codes only give indication of the actual
status of the trade against the contra-party. The
above position codes (BOX, FREE, SFK, F/D, F/R) are

15 all "True" position codes. A true position code
gives some indication of the actual status or
physical location relevant to the trade.

Position codes can also be used to indicate memo
positions. In the present invention, two types of

20 position codes are associated with customer
positions: a "true" position and a "memo" position.
A memo position indicates how customer security
positions (held in streetside name) have been used
by the firm. Memo positions such as "FREE" (free

25 for firm use), "SEG" (segregated from firm use),
and "SFK" (safekeeping in firm vault) all indicate
how customer positions are utilized by the firm.

         i.   Position and Balance Tables

    As implemented for a securities trading

30 organization, the position and balance files 62
comprise seven different files, as depicted in FIG
17. For purposes of a preferred embodiment of the
present invention where the transaction processor

-88-

is implemented in a three-tiered hardware
environment of minicomputers, microcomputers and
mainframe, all product positions are maintained
separately by trade date and settlement date.

5   A set of trade date position tables 602, 604 and
606 contain position data on executed transactions
whose settlement date does not yet equal the
current date.

By trade date, firm account trade date file 604
10 contains information on executed transactions, as
of trade date (i.e. when the trade is executed), by
firm accounts. A customer account trade date file
602 contains trade date data on security positions
by customer account.

15 A location trade date file 618 contains information
on executed transactions as of trade date as to
streetside location of the products (e.g.
securities) involved in the transactions.

A balancing function is inherent in the position
20 and balance module through the netting of the
customer and firm account positions on the one
hand, and the location positions on the other.

Using the example of a securities trading firm, the
firm and customer account positions represent
25 account ownership of the securities while the
actual location of the securities represented by
those firm and customer account positions are
recorded in the location tables. Security location
can be in many "street side" locations established
30 by the firm, such as the firm vault or the DTC
(Depository Trust Company). Thus, as product

ownership and security location define a security
position, account ownership balances against
security location. Customer and firm account
positions, by trade date (and settlement date)

5   should balance against location account position
(by trade date and settlement date). Because
different modules of the transaction processor of
the present invention supply data to update
different P&B tables (i.e., account-side/location-

10   side, as will be described) the table structure of
the position and balance module 64 of the present
invention, presents an innovative way to track
transactions information discrepancies. Where
account-side entries fail to match location-side

15   entries the P&B module 64 creates "dated-break"
records as will be described more fully below.

Referring to the tables in FIG 17, executed
transaction data is also maintained by settlement
date in balancing account-side and location-side

20   data tables. A customer account settlement date
table 608 maintains settlement date transaction
data by customer account. A firm account
settlement date table 610 maintains settlement date
transaction data by firm account. Those firm and

25   customer settlement date table entries can be
balanced against entries in a location settlement
date table 612. As will be described more fully
below, a product account memo table 614 contains
entries that link, by settlement date, customer

30   memo positions to physical locations.

FIG 18a-b depicts possible record entries in a
typical settlement date position and balance file
for the present invention, as implemented for a
securities trading organization. FIG. 18a depicts

typical balancing entries on the settlement date
firm account table 610 and the settlement date
location table 612. Entry 610a shows that the
trading firm holds $5,000,000 worth of United

5 States Treasury Bills in its account "TRD-ACCT-A."
In the present example, the key to the firm
settlement date position file 610 is a company (the
entity which executed the transaction, a product
(T-Bills), and an account. In addition to the key

10 information, a typical firm account record (trade
date or settlement date) would include the quantity
of securities held.

The firm account table records (both trade date and
settlement date) carry no position code. Only

15 ownership at the T-Bills and the account (TRD-
ACCT-A) is position information indicated by record
610a.

The locations account table entries 612 should
balance against security position held by the firm

20 in 612 its firm account 610. Typical entries 612
and 612 to the settlement date location table show
that $5,000,000 in United States T-Bills, dated
1/18, are held in two location accounts at "IRV"
and "MHT", and they are in "FREE" position, ready

25 for firm use. The keys to the location file are
company (firm entity which executed the trade),
contra party/physical location account (e.g., IRV,
MHT,), Merril position code and product. In
addition to the identifying keys, the location

30 account table 612 will contain other information
such as product quantity.

The balancing of the settlement date firm account-
side and location-side record entries is indicated

by the arrows in FIG 18. Account and location
records are matched by company, product position
and quantity. A unique feature of the present
invention its multi-entity balancing capability.

5   Using the company key, account and location records
can be balanced by the entity that performed the
transaction such as a trading firm's New York or
London office.

Where account entries fail to match corresponding
10 location entries, the position and balance module
64 will create a "break record" entry in the
location file (in both trade date and settlement
date cases). The break record would contain a
"BRK" position code and the date of the misbalance.
15 Dating record breaks is advantageous because data
discrepancies do not proceed to the next day
unmarked. The dating permits a stock researcher to
locate the cause of the break with better
efficiency.

20 FIG 18b shows typical balancing table record
entries in the P&B customer settlement date table
608, the location settlement date table 612, and
the product account memo table 614. The entries in
the customer account settlement date table 608a and
25 608b show the holdings of customer A by account:
1,500 shares of EXXON allocated to a margin account
608a and 3,000 shares of EXXON stock allocated to a
cash account 608b. The keys in the customer
account tables (trade date and settlement date) are
30 company, product, and account (sub-account). A
quantity code is also indicated, but no position
code. In the example, sub-type on the account-side
shows only that the customer has taken ownership of
the securities in its margin and cash accounts.

-92-

The settlement date location table entries 612c and
612d depict street side location positions in EXXON
stock. 4,500 EXXON shares are held in a DTC street
side account. However, two different position
5   codes apply to the street side holdings, in this
example, in accordance with securities industry
trading rules. 1,000 shares are "FREE" for firm
use (612c). While 3,500 are not available for firm
use, as they have been segregated "SEG" to fulfill
10 regulatory requirements. As a matter of securities
industries practice, when securities are purchased
on margin for a customer account, but in the firm
name, a certain portion of the securities held can
be used by the firm for loan collateral and other
15 purposes. However, a portion of these securities,
by regulation, must be held to cover the customer's
margin buy. The "SEG" code is used in location
table record 612c to show the status of the 3,500
shares of the EXXON stock in the street side "DTC"
20 location.

The customer account and location tables in FIG 18b
balance as indicated by the arrows. However, the
present invention also comprises customer account
position memo file entries 614a-c to further link
25 settlement date customer account and location data,
beyond balancing offsetting record entries.

In the present invention two types of positions are
associated with customer positions: a "true"
position and a "memo" position. Each record in the
30 customer account table 608 (e.g. 608a and 608b)
contains the "true" position. In the example of
FIG 18b the true position is that customer A has
taken ownership of 4,500 shares of EXXON, allocated
by 1,500 to a margin account and 3,000 to a cash

account. The present invention also associates
internal "memo" positions to the customer account
holdings. The memo position relates the customer
holdings to a physical location position code (e.g.
5   TRANF, SFK, REORG). For example, securities
purchased by a customer with cash cannot be used by
the firm for loans and other purposes. However a
certain portion of securities purchased by a
customer margin can be used by a firm for loans and
10   other purposes. The rest must be segregated.
Thus, apart from the true position codes, memo
position indicate how the firm may use customer
securities.

The description key for records in the customer
15   account memo table 614 (e.g. 614a, 614b or 614c) is
a combination of company, customer, account (and
subaccount), position code and associated account
field. The important field on memo table records
is the associated account field. The associated
20   account field provides the link between memo 614
location 612 position records, because the
associated account field in the memo table 614
contains corresponding entries from the account
field in the location table 612. Using the memo
25   position table 614 with the associated account
field provides a flexible way to match customer
account record entries with location table entries.

                ii. Input to the Tables

Referring to FIG 17, the P&B module 64 comprises a
30   position and balance manager module 600 (P&B
manager) a P&B trade processor module 616; a P&B
balance activity module 624; a P&B trade reference
table 618; a P&B transaction activity table 620;

and a P&B activity table 622. It is the P&B
balance activity module 624 that performs the
"break record" balancing of account-side and
location-side records by trade date and settlement
5   date as described above. The P&B balance activity
module, performs in batch at the end of the day
while the P&B tables are updated by other
components of the P&B module on-line during the
business day.

10   All updates to the P&B tables 62 are made by the
position and balance manager module 600. The P&B
manager module 600 receives data updates initiated
by many different transaction processing modules
(e.g. trade entry modules 48), and makes the P&B
15   file updates based on the type of transaction data
entered. Because the many different modules handle
different aspects of the transaction process and
all feed data to P&B module 62, the P&B tables are
a true central storage place for transaction
20   position data.

Following the example of the present invention as
implemented for a securities trading firm, the
trade entry modules 48 provide data on newly
executed trades, "as of" trades, and cancelled or
25   corrected trades. As will be further described
below, the P&B trade processor modules 616 first
receives raw trade entry data and, after processing
it, sends it to the P&B manager 600 to create the
appropriate P&B table 62 updates. The P&B module
30   64 processes executed trade data on customer
account transactions as it is sent from the
executed trade modules 48 - on-line during the
trading day. For customer trades having a
settlement date in future of a trade date, the P&B

manager module 600 updates the trade date, customer
602, and trade date location tables 606 as is
needed. For customer "as of" trades, cancel and
corrected trades, and cash trades (which settle on
5   the date the trade is executed), the P&B manager
module 600 creates appropriate updates the customer
account 602 and 608 location 606 and 612 and
customer account memo table 614 by trade date and
settlement date.

10 Transaction information concerning firm accounts is
received from the trade entry modules 48 by the
firm inventory module 66. It is the firm inventory
module 66 that forwards firm account information to
the P&B manager module 625. Based on transaction
15 data sent by the firm inventory module 66, the P&B
manager module 625 will update the appropriate firm
account table 604 and 610 and location table 606
and 612 by trade date or settlement date or both
(in the case of "as of" cash, and cancel or correct
20 transactions).

The clearance module 52, as will be described in
detail below, tracks and records all settlement
activity related to transactions. In a batch
process, the clearance module 52 provides the P&B
25 module 64 with "settlement date set-up" information
for all executed transactions that are due to
settle during the next trading day. The P&B
manager module 600 uses the transaction information
provided by the clearance module 52 to remove
30 customer 602, and location 606 table record entries
by trade date and establish customer 608, memo 614,
and location 612 positions by settlement date.
(The firm inventory module 66, described below,
provides "settlement date set up" input for the P&B

firm account files 604, 610). In addition, the
clearance module 52 sends to the P&B module 64 on-
line, during the trading day, transaction
information of security receipts and deliveries.

5   The P&B manager module 600 processes these events
as they happen to update settlement date location
position codes 600.

In a batch process, the dividends, interest and
redeemables (DIR) module 70, as will be described

10 below, sends transaction position information
concerning stock splits. In batch process, the P&B
manager module 600 updates the appropriate customer
(602 and 606), firm (604 and 610) and location (606
and 612) tables based on the entitlement

15 information sent.

The customer account module 68 sends to the P&B
module 64 transaction information on the free-
receipt or delivery of customer account stock and
other location changes to customer account stock.

20 FIG 18c-d depicts typical P&B table 62 updates made
on line, based on transaction information provided
by the clearance 52 and customer account 68
modules.

FIG 18c depicts three transaction records sent by

25 the clearance module 52 to the P&B module 64,
during batch "settlement date set-up" processing.
Record 630 depicts a firm sell to customer A of
1,500 IBM shares that was purchased on a margin
account. Record 634 depicts a firm sell to

30 customer A of 3,000 IBM shares, purchased with
customer A's cash account. Record 632 depicts a
firm buy of 4,500 IBM shares, presumably to cover

the customer A account transactions, as all
transaction are to settle on the same day.
Attached with each record is a transaction code
sent by the clearance module 52.

5   Based on the transaction code, the P&B manager
module will create the following updates to the
settlement date customer account 608, product
account memo position 614, and location 612 tables:
using record 630 and 634 the P&B manager module 600
10   places a record in the settlement date P&B customer
table 608c-d and the product account memo position
file 614e-f indicating the ownership of 1,500 IMB
shares in customer A's margin account and 3,000 IBM
shares to customer A's cash account. P&B manager
15   module 600 uses record 632 to create an entry in
the settlement date location table 612e.

The entries made in the P&B tables 62 and the
positions used for those entries, are determined by
the transaction code entry on each record fed to
20   the P&B manager module 600. A single transaction
input can cause the P&B manager module 600 to make
many P&B table 62 updates. The P&B manager module
600 uses the transaction code to access a P&B
activity table 622, to determine the P&B table
25   updates. Associated with transaction codes in the
P&B activity 622 are indicators of the P&B table
updates that need to be completed and the relevant
position codes.

Notice that in this example as of the night before
30   settlement, date the position codes on the customer
account memo tables indicate that customer A's IBM
stock are free for firm use on the eve of the
settlement date while the true location position

entry indicates that the securities have not yet
been received (failed to be received "F/R"). That
position code configuration is a normal
implementation of a securities industry practice
5   where legal title is taken to securities as of
settlement date regardless of whether the
securities are actually received. There is no
associated memo position for a "F/R" position,
because the "F/R" position does not indicate a
10 physical location.

FIG 18d depicts further P&B Table 62 updating input
that arrives on-line from the clearance module 52
during the next business day, and input that
arrives form the customer account module 68 at the
15 end of the next business day.

As securities set for settlement clearance are
actually received or delivered, the clearance
module 52 sends records such as record 636. Record
636 indicates that 4,500 IBM shares were received
20 as planned from "MERRILL", and they are located
(either physically or electronically) at a firm
account at the Depository Trust Corporation (DTC).
Using that information and the transaction code
carried on the clearance input 636 the P&B manager
25 module 600 adds to the settlement date location
file 612 record 612f, cancelling the previous F/R
record for 4,500 IBM shares 612e. In addition, a
new record 612g is added showing the stock receipt
and its new location at DTC. The position
30 indicator, "FREE", shows they are available for
firm use. Again, there is no associated account
position for the "FREE" position, because it does
not indicate a physical location.

In a batch process, described more fully below, the
customer accounts module 68, determines, based on
securities industry practice rules, the amounts of
customer account securities that must be segregated
5   and therefore cannot be used for firm purposes such
as loans. That transaction information (records
638, 640) is sent to the P&B module 64 to update
the P&B table 62.

In the example of FIG 18d, record 640 indicates
10   that all of customer A's cash purchase of IBM stock
must be segregated from the firm usable securities.
Using the transaction code, the P&B manager module
600 creates entries 614i and to cancel the "FREE"
memo position entry 614f and also add record 614j
15   showing that 3,000 IBM securities have been
segregated (SEG) from the associated account DTC.
P&B manager module 600 also creates similar updates
to the location table 612, offsetting the "FREE"
position record 612g with free record 612i and
20   adding SEG record 612h. The SEG position code, as
used in the location tables, is indicative of a
physical location, thus the associated account
"DTC" is placed in the memo entries, linking the
memo and location entries.

25   The margin module 64 also sent record 638,
indicating that 500 of the 1,500 IBM shares in
customer A's margin account must be segregated from
firm use. Using the transaction code, the P&B
manager 625 creates updates to the customer memo
30   file 622. Record 614g offsets 500 IBM shares from
the "FREE" position record 614e. Record 614h
indicates that 500 shares of IBM shares have been
placed in SEG at the DTC account.

-100-

The P&B manager 600 also updates the location table
612, using information from record 638 update
record 612k offsets 500 "FREE" EXXON shares from
the 4,500 shares indicated by record 612g. In
5 addition, entry 612j indicates that 500 shares of
EXXON stock in the DTC stock have been segregated.

        iii. P&B Manager Module Process Flow

Referring again to FIG 17, that figure illustrates
the overall system flow of the P&B module 64. As
10 discussed above, different transaction processing
modules (i.e. trade entry modules 48, clearance
module 52, firm inventory module 66, customer
accounts module 68 and DIR module 70) send
different transaction data to the P&B module 64.
15 Each different type of transaction data sent to the
P&B module 64 will cause one or more position
activity updates to the P&B file, as was shown in
FIGS 18c-d.

To send data to the P&B module 64 each feeding
20 module sends with the transaction, a transaction
code, as was seen for example on records 632, 634
and 636 in FIG 18b.

Transaction codes are table driven for trade
related processing in the trade entry modules 48,
25 clearance 52 and firm inventory 66 modules in the
present invention. The codes are maintained in a
trade reference table 603. Based on the different
facts of the transaction (e.g. firm or customer
transaction, Buy or sell), each module accessing
30 the trade reference table 603 (i.e. clearance
module 32, firm inventory (during settlement date
setup) 66, the P&B trade processor (for the trade

entry module 48)) will forward to the P&B manager
600 its transaction data together with the
transaction code located in the trade reference
table 603. For non-trade related transaction input
5   such as that from the customer account 68 and DIR
70 modules, there are so few codes needed to be
maintained that they can be more efficiently
maintained by not incorporating them into a table.

The P&B manager 600, using the data and the
10   transaction code, creates from one to many position
activity updates to the P&B files 62 depending on
the transaction code. To determine the appropriate
activity updates, the P&B manager module 600 uses
the transaction code to reference the P&B activity
15   table 622. The P&B activity table 622 lists the
appropriate P&B activity updates and corresponding
position codes for each transaction code entry.
Using the data received and the activity
references, the P&B manager 600 creates activity
20   records to update the P&B tables 64.

As an audit trail, those records are time stamped
and stored in a P&B transaction activity table 620.
Generally, when the P&B manager 600 makes the P&B
table entries 62 it also accesses the P&B
25   transaction table 620, marking each update as
completed, and return acknowledgement to the
feeding modules such as the trade modules 48 and
the clearance module 52. However, variations of
this procedure exist as will be described below.

30   For purposes of a preferred embodiment of the
invention components of P&B module would exist in
both the minicomputer and mainframe computer
environments. (See 3 and 36, FIG 3). The process

flow for such preferred implementation is depicted
in FIGS 19a-b.

FIG 19a depicts the P&B module process flow where
transaction processing modules, (such as the trade
5   entry modules 48, and firm inventory 66), feed
transaction data to the mainframe resident P&B
module 64.

The trade entry modules 48 notifies the P&B system
to the mainframe environment 36 by the store and
10   forward system 43 and 41. The trade entry modules
48 write transaction data to the store and forward
log 43 and notifies the (minicomputer resident) 34
store and forward module 42 using the notifier
module 380 as described above. The mainframe
15   resident store and forward module 41 receives the
trade data, writes to massive trade entry storage
table 44, and, using the store and forward
notification module 46, notifies the mainframe
environment modules, firm inventory 66, and the P&B
20   trade processor module 616 of the massive storage
table reference to the new data.

If the transaction data involves customer account
transactions the P&B trade processor module 616
(like clearance (for setup) 52, firm inventory
25   (setup) 66), will take the data and invoke a P&B
trade reference table access module 704 to
reference the appropriate transaction code from the
trade reference table 618. The other mainframe
environment 36 transaction processing modules (DIR
30   70 and margins 68) must know the transaction codes
associated with the business event.

Once the transaction code is returned, the data and

code is sent to the P&B manager module 600 to build
the appropriate activity records and update the P&B
tables 62.

The P&B manager module comprises a number of sub-
5   modules to obtain the activity data based on the
transaction code. A P&B sub-manager module 710
receives transaction data and the transaction code
and forwards the code to a format P&B activity
module 712.

10   That format module 712 sends the information to an
activity and position code accessor module 714
("accessor module"). The accessor module 714 uses
the P&B activity table 622 to determine what P&B
table updates should be completed based on the
15   input information.

Appendix VII contains examples of position code
table entries that are used in creating different
activity updates. The accessor module 714 uses
account sub-code (01, 02, etc.) that has been sent
20   from the input to obtain a position code for all
customer trades. The accessor module 714
accomplishes its references to the P&B activity and
position table 622, using a table access module 720
designed to traverse the activity and position
25   table 622. The accessor module 714 formats the
transaction activity record and returns the records
to the format P&B activity module 712.

The format routine 712 next sends a request to time
stamp the P&B table activity updates to a P&B
30   transaction time stamp routine 724.

The time stamp routine 724 returns a current time,

an environment indicator number, and a sequence
number for all activity records that will be
created. The environment indicator shows the
origin of the data that created the P&B table

5   activity updates. Each update record to be created
will also carry a sequence number assigned by the
build transaction number 724. The combination of a
transaction number and a sequence number makes each
update unique. The format routine 717 returns a

10   list of all updates to be created and their unique
ID numbers to the main P&B sub-manager 710.

The P&B manager 600 also comprises a number of sub
modules to create the activity updates and write
them to the P&B tables 62.

15   The P&B sub-manager 710 sends the set up data to an
update P&B file module 726. The update P&B file
module 726 creates the P&B activity updates and
sends these records to the appropriate P&B tables
such as the trade date or settlement date customer

20   account files (612, 616). Each file has an access
module 732 to complete the actual updating.

If the update is successful, the update module 726
sends the activity record with a valid return code
to an update transaction activity log module 734.

25   The P&B transaction activity log 620 tracks all
valid P&B database updates.

If the update P&B files modules encounters an
invalid return code, an error code and file name is
sent to the update module 726 and the update

30   transaction activity log file module 734 returns
the record to the P&B sub-manage module 710, which
writes the error record to the transaction activity

table 620. An error code is also returned to the
P&B sub-manager module 710 or then to the mainframe
environment feeding modules (i.e. firm inventory
66, DIR 70, customer accounts 68, or P&B trade
5 processor 601). The feeding modules internally
handle invalid return codes and will resend the
trade data. A record with an invalid error code is
not forwarded to the P&B transaction activity file
620. A process input module 738 tracks all file
10 access errors in an error report file 740.

The update routine 726 repeats similar steps for
each P&B table 62 update that is to be created.

When all activity records are successfully written
to a P&B and activity file, a valid return code is
15 sent by the P&B sub-manager module 710 to the
mainframe feeding modules.

FIG 19b depicts the processing flow for transaction
related data processed in the minicomputer
environment 34.

20 A transaction processing module in the minicomputer
environment 32 such as the clearance module 52
accesses a minicomputer resident copy of the
reference table 618 and sends transaction code and
related data to a minicomputer resident P&B module
25 65. The P&B module 65 comprises a number of sub-
modules to create the activity updates. A
minicomputer-resident P&B sub-manager module 775
sends all non-trade information to a minicomputer-
based edit checking module 774. The edit checking
30 module 774 uses general reference database
information from the customer account reference
database 206, the product master reference database

204, and product classification trees 214. (See
FIG 7). Database access modules 782, 784 perform
the database traversals. The edit checking module
774 validates account number and product number and
5   checks for required fields. If an invalid code is
returned, the minicomputer resident P&B sub-manager
module 775 returns the code to the feeding system
and processing terminates. If the module returns a
valid return code, processing continues, and P&B
10 sub-manager module 775 sends the P&B transaction
code and trade data to a minicomputer resident,
format activity module 772.

The minicomputer resident format activity module
772 sends the transaction code and input data
15 information to an activity and position code
accessor module 776 ("accessor module"). The
access or module 776 uses the transaction code to
access a minicomputer-resident P&B activity table
622 and determine the P&B table updates that must
20 be created. The activity and position code
accessor module 776 uses the account sub-type (01,
02, etc.) carried on the trade to obtain the
appropriate position code for customer trade data.
The minicomputer-resident activity and post
25 accessor module 776 performs the table accesses 600
using a table access module 782.

The format P&B activity module 772 also sends a
time stamp request to a minicomputer-resident P&B
time stamp module 784. The time start module
30 creates a number that will be assigned to each of
the P&B activity updates.

The format module 772 creates the P&B data update
records and returns all records to the P&B sub-

manager module 775. The minicomputer resident P&B
sub-manager 775 writes all transaction activity
records to a minicomputer resident P&B activity
file 620 and minicomputer-resident store and
5   forward log 43. The P&B sub-manager module 775
then returns a successful update return code to the
transaction processing module such as clearance 52.

The store and forward module 42 sends the
activities to the mainframe environment store and
10   forward module 41, which writes the minicomputer-
resident log records to a mainframe P&B transaction
activity table 620, and invokes the store and
forward notifier module 46 to inform the mainframe
resident P&B manager module 600 of P&B activity
15   updates that have been generated on the
minicomputer. Notice that in FIG 17a as records
are processed in the mainframe environment,
activity records are stored in the transaction
activity table 620 -- as the P&B tables are updated
20   -- with their validity codes already indicated. In
the present example of FIG 19b, where activity
records come initially from the minicomputer
environment, they are immediately placed in the
transaction activity table 620, without their
25   validity codes indicated and before they are
written to the P&B tables 62. The P&B sub-manager
module 710 retrieves the corresponding records from
the P&B activity table 605. The P&B sub-manager
710 records and forwards the updates to the P&B
30   update module 726. The update module 726 sends
update records to the appropriate P&B table (e.g.,
the firm account settlement date table 610) using
one of the access modules 732.

If a valid return code is returned by the update

module 726, the P&B sub-manager 710 returns to the
transaction table 620 and updates each activity
recorded as validly received.

If an invalid return code is encountered by the
5 table accessor modules 732, the return code and
file name are returned to the P&B sub-manager
module 710. This record is written out to the P&B
error report file 740.

The P&B sub-manager module repeats the steps for
10 all activity records associated with the
transaction.

4. Firm Inventory Module

The firm inventory module 66 as depicted in FIG 3
of the present invention, updates the P&B tables
15 62, as described above, and in addition, maintains
account lot lists for liquidating securities held
in firm accounts. Utilizing the present invention,
firm security position holdings are ordered and can
be liquidated using a first-in-first-out (FIFO) or
20 other liquidation method.

The firm inventory module 66 calculates net profit
and loss for securities held in firm accounts on a
trade date and settlement date basis, according to
lot liquidation method, and further, it calculates
25 cost of carrying securities from settlement date to
liquidation date.

FIG 20 depicts the process flow of the firm
inventory module 66. As implemented in the
exemplary three-tiered hardware environment of PC,
30 minicomputer and mainframe, the modules comprising

the firm inventory module 66 reside in the
mainframe environment 36.

The primary data inputs to the firm inventory
module 66 are: 1) executed trade records from the
5   trade entry modules 48; and 2) entitlement records
from the dividends interest and redemptions module
(DIR) 70. The main input feed is the trade entry
modules 42; from it firm inventory receives
executed buy and sell records, "as of" trades and
10   cancelled and corrected trade records. An "as of"
trade is a trade entered into the system on the
current date but effective "as of" an earlier date.
The DIR module 70 sends stock and cash entitlement
information, such as stock and cash dividends.

15   The minicomputer based-trade entry module 42 and
mainframe based DIR module 70 write input records
to corresponding massive storage tables 808, 810
resident on the mainframe (see also FIG 3, 44),
from which the modules of firm inventory will
20   access the data. For trade entry records, the
store and forward module 42 takes the records from
the minicomputer-resident store and forward log 43
and writes them to the massive storage table 808.
The mainframe resident store and forward
25   notification module 46 receives a reference number
to the massive storage table record. The reference
number is sent to an input queue 818 and received
by a firm inventory access module 820.

A mainframe-resident transaction processing
30   modules, such as the DIR module 70, writes
information directly to its storage table 810.

In a batch process, at the end of the processing

-110-

day, the firm inventory access module 820 reads the
data records 808 or 810 and simultaneously passes
them to a firm inventory P&B update module 822 and
a trade lot processing module 824.

5   The firm inventory P&B update module 822 processes
the input data by invoking the P&B manager module
600 and sending it relevant data and a transaction
code. The P&B manager module 600 receives all firm
principal transactions from the trade entry modules
10   42, as well as data from the DIR module 70 on
scheduled security entitlements related to firm
principle trades, such as stock share dividends.
Cash entitlement records from the DIR module 70 are
not recorded in position and balance files 62.
15   Those entitlements, however, are tracked by the
firm inventory module 66 to determine net profit
and loss and to determine the cost of carry.

A trade lot processing module 824 applies the trade
entry and DIR data to files grouped as liquidation
20   schedule lots 806 using a predetermined liquidation
method. Security positions can be liquidated on a
FIFO or other basis, such as maximize/minimize
profit, or maximize/minimize long term capital
gain. A user can choose the liquidation method for
25   a security position when entering the trade. A
default liquidation method, such as FIFO, is
otherwise utilized.

Transaction records that "build" in an account
position, such as a stock purchase, are added to
30   open position lot files 826 and 827. Transaction
records that subtract from or "close" a position,
(such as a stock sell), are maintained in closed
position lot files 828, 829. Cancel and correct

lot files 830 and 831 are used to alter corrected
open or closed lot entries as well as create
cancelled lot entries. Open lot 826, 827, closed
lot, 828, 829, and cancel lot 830, 831, files are
5 maintained by trade date and settlement date.

The firm inventory module uses the figuration
routines 212 from the general reference databases,
described above, to and access information stored
in the product master database 206, and to
10 calculate realized profit and loss. Realized
profit and loss records are maintained in, the
closed lot files 828, 829.

The present invention maintains liquidation
processing order even against "as of" and cancels
15 or correct processing of the lot lists. "As of"
processing is described more fully below using FIG
23b. However, a lot unwind/rewind module 894
performs "as of" and correction processing. The
unwind module rolls the open and closed lots 826,
20 828 to the point of the "as of" update or
correction. The rewind module 894 processes the
trades subsequent to the correction. The rewind
module 894 accesses past input from the massive
storage tables 808, 810 to reprocess the open and
25 closed files.

FIG 20b desribed below, depicts the lot record
movement in handling an "as of" posting to FIFO-
based open and closed lot files.

Referring to FIG 20, the present invention also
30 comprises batch processing modules 858, 863 to
access P&B firm account tables 604, 610 to create
settlement date lot positions. At the end of each

-112-

business day for a branch of a securities trading
firm, the settlement date batch processing module
863 reads the massive storage tables 808, 810 and
creates settlement date lot file entries 827, 829
5   for each security that is due to settle during the
next day. The new settlement date records are
entered into the settlement date lot processing
files 827 by the chosen liquidation method (e.g.
FIFO).

10 In addition, the P&B settlement date update module
858 creates, in batch, P&B table 62 updates for the
securities held in firm trading accounts that are
expected to settle ("settlement date set up"). At
the end of the business day for a branch of a
15 securities trading firm, the P&B settlement date
update module 858 reads the firm account trade date
file 610 in the P&B database 600 and creates firm
account settlement date file 614 record entries for
all the trades expected to settle on the following
20 day.

During end-of-day batch processing of the firm
trade date P&B file 610, the P&B settlement date
update module 858 records accumulated unsettled
sales. As financing is based on settlement date
25 position, less risk and better financial terms are
available for a broker when the positions held have
already been sold.

The batch settlement date module 858 also triggers
a batch process module 864 to compute the cost of
30 carry for all records in the settlement date
position and balance table 610.

The cost of carry is the amount of money it costs a

securities trading firm to hold the securities,
instead of investing its money on other products.
The cost of carry is a sum of any financing charges
incurred in borrowing money to pay for the
5   securities plus the lost interest a securities firm
could have made by loaning the money instead of
purchasing the securities. The cost of carry
computation module 864 calculates those charges for
all records in the settlement date positioned
10 balance table 610 allocates the costs against the
profit and loss accounts.

Cost of carry is determined by taking all settle-
ment costs and applying a collateralization
formula. A money store module 868 calculates
15 collateralization based upon such figures as the
previous days bank loan rates. Cost of carry
records are written in a net carry file 870,
allocating the carrying cost to specific firm
trading accounts. To determine what calculation
20 formula to use, the cost of carry module employs a
product classification tree 214, and the product
master database 206 for calculating treasury and
finance category information.

For failed trades that do not clear on the
25 projected settlement date, the cost of carry
account file 870 is updated by the clearance system
as will be described below. A failed trade
processing module 867 of clearance credits the cost
of carry against each account for the days where
30 securities were not actually carried because the
trade failed to settle.

In addition, there is a linking module 878 used to
potentially offset some of the cost of carry. A

-114-

convertible arbitrage position linking module 878
links offsetting trades for convertible-type
securities in an attempt to identify valid
convertible arbitrages.

5   In addition to calculating the cost of carry, the
present invention also comprises additional modules
to make other firm account calculations. A
deferred earnings module 904 calculates deferred
earnings for discounted securities for users that
10 choose to realize profit and loss on a trade date
basis. The present invention comprises an
unrealized profit and loss calculation module 906
to track trade date and settlement date positions
and calculates on a sales against a cost basis.

15 Unrealized profit and loss totals are maintained in
an unrealized P&L file 907. Statistics such as
turnover, and aging are tracked by a position
statistic module 908. Statistic totals are stored
on the P&B firm account tables 604, 610.

20 On a settlement date basis, other modules calculate
cost accretion for discounted securities, 910,
settlement date position earnings 912, as well as
the cost of carry described above. Profit and loss
is aggregated on a monthly 916, yearly 918 and life
25 to date basis 920.

A report processing module 900 accesses the
position earnings files 916, 918, 920, 855, the
position statistics file 909, the unrealized P&L
file 907, the price accretion file 911 and the cost
30 of carry total file 870 to create firm inventory
reports 901.

In addition to the report output generated, firm
inventory also updates the general ledger interface
system described below and the money store system
tables 868.

5   The present invention also comprises a manual
adjustment module 902 that permits users to make
manual corrections to the firm position lot files.
A user can view the firm inventory lot files 806,
using an on-line, front-end module 904. Manual
10   adjustments to either the position file or the lot
files will trigger the lot unwind/rewind module 894
in appropriate circumstances to change lot order.

The viewing module 904 comprises a function to
examine the open lot files 826-827. The viewing
15   module 904 has the ability to view the open lots
using different liquidation strategies, such as
FIFO.

An example of lot processing file action is shown
in FIG 20a. Suppose the trade entry module sends
20   records of IBM stock purchase activity for Account
1 to the firm inventory module. Initially, there
are three records: two buy transactions 832, 834
and a later sell 836. Trade 1 832 and then trade 2
834 are entered into a trade date, FIFO open
25   position lot file 826. Trade 3 836, the sell trade
input 836 will liquidate the built position 838 +
840 in IBM for Account 1. The information from
trade 3 is used to liquidate the securities from
the open lot file 826 on a FIFO basis. Entry 838a
30   in the open lot file 826a shows the results of the
update. An entry 842 is made to a FIFO trade date
closed lot file 828 which shows the liquidated
stock.

FIG 20b depicts the lot record movement in handling
an "as of" posting to FIFO based open and closed
lot files.

FIG 20b, shows data obtained from the trade entry
5   modules 48 for three trades: T1 832; T2 834; and T3
836. The listing at 826 and 828 reveal partial
file listings for the trade date open and closed
position lot files. Trade record, T4 890 arrives;
it has an "as of" effective date prior to the
10   records previously processed. Referring back to
FIG 20, the firm inventory unwinding module 894
reviews the closed lots, processing back to the
effective date of the "as of" record, and
reprocesses the trades from that date by inserting
15   the "as of" record in its appropriate lot sequence.
As mentioned above, the massive storage table 808,
810 in the mainframe environment 36 contains all
data input from the trade entry modules 42 and DIR
70. The unwind/rewind module 894 reviews the trade
20   entry module data table 808 until the "as of" can
be placed in proper sequence. Then, the
unwind/rewind module 894 rebuilds the open trade
lot file 826 adding the "as of" trade in its proper
sequence then re-adding the subsequent trades.

25   A batch reconciliation module 898 matches the lot
liquidation files 806 with position and balance
database 600 files. The reconciliation module 898
submits any breaks occurring to a report facility
900 that generates reports. Researched breaks can
30   be manually adjusted using the manual adjustment
module 902.

5.   Customer Accounts Module

-117-

The customer accounts module 68 of the present
invention comprises functions of monitoring
customer accounts, performing customer accounting
functions, logging customer activities and
5   generating periodic customer statements. As
implemented for a securities trading organization,
the customers of the organization can hold two
basic types of accounts, cash and margin. There
are, however, different sub-type classes of margin
10   accounts such as a short account. For margin
accounts, the task of monitoring the buy and sell
activities of the customer and producing customer
balances is complex.

For example Regulation T, promulgated by the Board
15   of Governors of the Federal Reserve, places
requirements on the amount of funds available to
pay for security transactions. Tracking special
memorandum account funds "SMA" is a method of
keeping track of the customer account funds
20   available to be applied either to meet either
Regulation T's "initial margin requirements" on new
transactions or customer requests for withdrawal of
funds. As shown in Appendix VIII, many different
transaction inputs affect SMA calculations. The
25   basic rule is that any activity that changes a cash
or security position impacts SMA calculation.
However, activities that increase debit balance,
such as debit interest, does not impact SMA. As
implemented for a securities trading organization,
30   the customer accounts module 68 comprises software
to implement government regulation and firm imposed
constraints on SMA calculation. Where SMA becomes
negative because of trade related activities, a
notice such on a "Fed Call" or a "Reg T Call" is
35   sent to the customer.

In addition, the customer accounts module (68 as
implemented for a securities trading organization)
comprises modules to calculate maintenance margin
balances. Maintenance margin is calculated to

5   determine if a customer is under-margined according
to the standards of the trading firm itself or an
exchange such as the New York Stock Exchange. If a
customer is under-margined a notice, such as a
"house exception" or a "NYSE exception", is issued.

10   Further, the customer account module 68 of the
present invention comprises modules to process
margin and customer account exceptions in addition
to the Regulation T and house maintenance notices.
The additional exceptions comprise the processing

15   of standard industry balance indicators including:
minimum equity balances; unsecured exceptions;
technical short exceptions; liquidation exceptions;
day trade exceptions; 90 day restriction
exceptions; excess available exceptions; customer

20   account record keeping exceptions, position and
balance exceptions and other exceptions that are
defined more fully in Appendix IX. The customer
account module 68 maintains all exceptions and, on
a daily basis, handles their aging or new issuance

25   as described below.

The customer account module 68 of the present
invention also comprises modules to debit and
credit interest due on customer accounts. Cash and
margin account cash balances are summed and

30   multiplied by an applicable margin interest rate to
derive margin interest. The prevent invention is
comprised to accrue debit and credit interest
nightly and post interest to the general ledger
module 72 and cash management system module 54 at

-119-

the end of monthly interest periods.

The basic systems process flow comprising the
customer account module is depicted in FIGS 21a and
21b.

5   The basic functions of the customer accounts module
     68 comprise: 1) a daily gathering of transaction
     events, which occurs both on-line during the day
     and in batch at the end of business; and 2) batch
     calculations processes to calculate margin
10   requirements and other customer balances.

                   i.   Activity Capture

     FIGS 21a-b depict the modules that function to
     collect, both on-line and in batch, daily customer
     account transaction records. An activity table 930
15   (located, as a preferred embodiment of the
     invention in the mainframe environment massive
     storage tables 44) is the storage area for
     transaction data pertaining to customer accounts.
     The customer accounts module 68 receives data from
20   a number of sources on-line, makes appropriate
     entries to the activity table 930 and in some cases
     sends customer information to the transaction
     processing modules.

     The transaction entry modules 48 notifies the
25   customer account module, (on a real-time basis
     using the store and forward modules 41, 42), of: 1)
     customer trades on trade date; and 2) cancelled
     customer trades entered after settlement date. For
     each notification, a post trades module 926 reads
30   the customer transaction from the mainframe
     resident executed transaction table (See 44, FIG 3)

and creates an activity record in the customer
account activity table 930. For cancelled customer
trades after settlement date, the post trades
module 926 will also send record of the customer
5   cancel to the cash management system CAMS 54,
clearance 52, and general ledger interface (GLI) 72
modules using customer account interface modules
927, 931, 932.

As monies for transactions related to customer
10   accounts are received by the CAMS module 54, record
of the receipt or payment of money by check or wire
is sent, (using the store and forward modules 41,
42), to the customer accounts module 68. A post
CAMS activity module 936 records the transaction
15   event in the customer account activity table 930
and, in addition, sends record of the wire receipt
to the GLI interface module 944, which forwards the
record to the GLI module 72.

In the personal computer environment 20, a customer
20   account front-end module 934 sends transaction data
to the customer account module 68. On a PC, an
activity update screen function permits customer
account operators input transaction data
concerning: 1) manually initiated payment
25   requests; 2) manually initiated account updates on
monies or securities; and 3) free receives of
securities to be used as margin collateral for each
of these inputs. A mainframe resident customer
account on-line module 933 will create an update to
30   the customer account activity file 930. For
manually initiated payment requests, the customer
on-line module 933 will send request data to a CAMS
payment request module 928, (as the CAMS module 54
integrates all actual cash movement in the present

invention) and to the GLI interface module 932 to
forward to the general ledger interface module 72.
Manually initiated updates of quantities of
securities and data on free receives against money
5   details are also forwarded in a similar manner by
the customer on line module 946 to the CAMS 54 and
GLI 72 modules. Free receive data and manual
account update data are forwarded to a customer P&B
interface module 929. The data is sent to the P&B
10   module 64 for processing with a transaction code as
described above.

In addition to the account update screen function,
described above, the customer account front-end
module 38 comprises an inquiry function to users to
15   view current customer balances. A "shadow posting
function" will show customer account activity
records 930 juxtiposed with current balance
information. In addition, the inquiry feature of
the front-end module 934 permits users to save
20   notes on customer accounts in a customer memo 971
and a customer status table 969.

In real-time, the DIR module 70 sends to the
customer account module 68 transaction data on cash
entitlements and security quantity entitlements
25   received on or after the date of payment. DIR
records can also be accepted on a pending status
basis. A post entitlements module 925 creates
record updates to the customer accounts activity
file 930. For cash entitlement records, the post
30   entitlements module 925 sends the update to the
CAMS module 54, (using the call minicomputer module
45 (see Fig 3)), and the general ledger interface
module 72, (using GLI interface module 932). The
post entitlements module 925 sends quantity

entitlement records to the P&B module 64, using the
customer P&B interface module 929 as described
above. In addition the DIR module 70 will send to
the customer account module 68 transaction data on
5 pending entitlements. In those situations, the DIR
Interface Module will post the entitlement record
on a pending status basis to the customer activity
table 930. Later, when the entitlement issues, the
customer accounts 68 module will update the CAMS 54
10 and GLI 72 modules.

The present invention also comprises links with
outside transaction data sources. Using the
example of a securities trading firm, customers can
transfer accounts from one broker to another using
15 the Automatic Customer Account Transfer System
"ACATS" 936. Also, trading firms often arrange
with intra-firm money market accounts to permit
customers to transfer funds between the firm and
outside firm money market accounts 938.

20 A post-ACATS module 935 receives and sends
information in batch on account transfers and ACATS
transmissions. The post-ACATS module 935 also
creates activity records and forwards corresponding
records to the GLI 72, CAMS 54 and P&B 64 modules,
25 as well as making updates to the customer activity
file 930.

A post money market account module 937 receives and
sends in batch fund transfer information from
outside money market sources 938. In addition to
30 updating the activity file 930, the post money
market module 937 sends records of funds transfer
to the GLI 72 and CAMS 54 modules in the manner
described above.

-123-

ii. Settlement and Margin Calculations
Exceptions and Interest

The margin calculate module 960 is a batch process
that accepts and processes all the activities 930
5   entered during the course of the day. Its process
flow is depicted in FIG 21b. The functionality
comprises modules to adjust account balances,
maintain regulatory requirements, summarizes market
value, calculate account interest, note exceptions
10   where necessary, and compile data for exception and
other reporting and notification.

The functional system flow is divided into four
steps:

1)   check each customer and
15   decides if there are any
activities to process; calculate
impact on SMA and house
maintenance margin levels; issue
trade-related exceptions;

20   2)   clear all trades whose
settlement date has arrived;

3)   review the customer tables
to note any exceptions that may
be applicable;

25   4)   accrue interest daily and
notify the GLI module 72 of the
daily accrual.

In addition to the transaction gathering modules
and the activity table 930 described above, the

customer accounts module 68 comprises an initial
margin module 942, a house maintenance module 944,
a margins settlement balances module 946, an
exceptions calculation module 948, a notice
5 authorization module 950, a debit/credit interest
module 954 and a reporting module 979. The
activity table records 930 are used primarily to
update a customer account balances table 965 and
customer sub-account balances table 967. The
10 customer account balances table holds account
balances by trade date and settlement date, as well
as totals for SMA monies needed to meet initial
margin requirements. The customer sub-account
balances table 967 holds trade date and settlement
15 date account totals by sub-account, as well as SMA
and initial margin requirements for customer
accounts as broken down by sub-accounts, such as
Customer A-"CASH" Customer A-"MARGIN" and Customer
A-"SHORT".

20 The initial margin regulation T requirements are
checked by the margin calculation module 942.

Initial margin requirements, or Regulation T
requirements, are stored and maintained in a margin
requirements table 962. The information will be
25 kept by product classification type and entered by
operators at the securities trading firm. This
margin requirement table is a PCS classification
tree (as in FIG 7, 214). Initial margin
requirements also depend upon price for equities,
30 ratings for bonds and time to maturity for U.S.
Government obligations which are retrieved from the
firm price management database 210. Firm initiated
overrides of the initial margin requirements for a
product, product classification, a particular

customer, or customer range are used by the initial
margin module and are stored in a margin
requirements override table 963.

To determine if a margin account activities are
5   able to pass the initial margin requirement of
Regulation T, the initial margin module 942
calculates the SMA for each customer account, in a
five step iterative process. If the calculated SMA
is less than the required initial margin required,
10 then a "Fed Call" notice must be sent to the
customer.

For each customer account, the initial margin
module 960 takes the transaction activities (from
the customer activity table 930) for that account
15 and updates the account and sub-account balances
965, 967.

The initial margin module 642 also computes SMA, by
beginning with the opening SMA (found in the
customer's account balance table 965) and nets with
20 all non-trade related activities that impact SMA
from the customer activity table 930. The initial
margin module 960 distinguishes between trade and
non-trade customer activity records 930, and it
determines whether a particular activity will
25 impact SMA by referencing an activity matrix table
960. Entries in the activity matrix table 960 (as
shown in Appendix VIII) describes the impact upon
SMA for all possible activities and whether an
activity is considered to be trade or non-trade
30 related. Non-trade related changes are first added
to the old SMA value.

With this figure, the initial margin module 942.

-126-

reduces any outstanding "Fed Call" notices for
additional payment. An exception table 970 holds
records of outstanding call notices and their
values.

5 Next the initial margin module 942 will net trade-
related activity records from the customer activity.
table 930 as determined by using the activity
matrix table 960. The trade related activity will
cause further adjustments to a customer account's
10 SMA.

This adjusted SMA value will be compared against
the initial margin requirements for the particular
transactions making up that account. The initial
margin module 942 determines the initial margin
15 requirements for a particular account by accessing
the margin requirements table 962 (in the manner of
accessing a PCS tree as described above). If there
is inadequate SMA to meet the initial margin
requirements, the initial margin module 942, (if
20 the customer has authorized the company to), will
automatically move enough money to meet the
requirement from a cash account 938 to the margin
account; and 2) initiate the appropriate GLI 72
update. After any movement of money if SMA is
25 negative due to trade related activities that
amount becomes the amount of Fed Call needed.

If a Fed Call notice is required, the initial
margin module 942 creates a request for a Fed Call
by placing record of the margin deficit in the
30 exception table 970.

Where a Fed Call is to be issued, the SMA for the
account is zero. If there is no Fed Call, the .

initial margin module 942 calculates "excess".
Excess is the difference between market value of
securities in the account minus debit balance and
initial margin requirements. If the excess is

5   greater than current SMA, the initial margin module
942 adds the difference to the SMA, otherwise the
current SMA is the closing SMA.

To determine securities market value, the initial
margin module 942 invokes a mark to market module

10 (MTM) 952. MTM 952 will value the securities in
the P&B settlement date customer account file 616.

In addition to the initial margin module 942, the
house maintenance margin module (MMM) 944 performs
calculations to determine if a customer account is

15 under margined. If a customer account is
undermargined, a "house" and/or other maintenance
exception (such as NYSE) is issued.

To begin, MMM 944 will total the trade date money
balances in the margin and short sub-accounts found

20 for each customer in the customer sub-account
balance table 967. To value all trade date
positions in each customer's margin and short
accounts, the MMM module 944 searches the P&B file
customer trade date position table 602 and values

25 the positions there, using data from the firm price
management database 210. MMM 644 will next
calculate the customer's house and NYSE maintenance
requirements based on trade date positions. The
house and NYSE requirement percentages are

30 retrieved from the margin requirement table 962 and
the override table 963.

The margin override table 963 contains priority

margin requirements for specific customer ranges,
specific customers, products, product
classifications or a combination of the above.

The customer settlement balance module (CSBM) 946
5   is invoked, at the end of trade settlement date, to
determine whether customer account trades should
clear or fail, based on current account status. If
the transaction is a customer buy, CSBM 946
deterines whether the customer account has
10   sufficient funds to make the purchase. If the
transaction is a customer sell, CSBM 946 determines
whether the customer has maintained a sufficient
position to complete the transactions. CSBM 946
accesses data on trade settlement activity from the
15   clearance main file 60, matching settlement
transaction records against customer account
balance records 965, 967.

From the comparisons, CSBM 946 initiates updates to
the customer account balance tables 965, 967 and
20   the P&B customer memo table 614. The customer
account money balances tables 965, 967 track an
assumed settled balance (assuming all transactions
donot fail) and a fail settlement balance. The
customer memo table 614 updates are forwarded (with
25   transaction code) to the P&B manager module 64.

In addition, if a trade is failing, in certain
circumstances an extension in time to settle the
transaction can be granted by a trading entity such
as the New York Stock Exchange. An extension
30   update module 942 determines whether an extension
can be applied for and if so, creates a record
update NYSE extension table 977 that will be
transmitted to the NYSE.

In addition to the Fed Call, house call and NYSE
exceptions, an exception processing module 948
checks the account balance files 965, 967 for other
possible customer account problems.

5   For example, a minimum equity requirement amount is
a user maintained field in the margin requirements
table 962. The exception processing module 948
compares this amount with the total equity within
the customer's account 965, which includes all
10 sub-account types. This exception is calculated
when a customer's position is increased in the
margin account by margin buys and short sells. For
margin buys, if the minimum equity requirement
amount is greater than the total equity, an
15 exception will be issued. The exception amount
will be the trade amount or the minimum equity
requirement amount whichever is lower. For short
sells, if the minimum equity requirement amount is
greater than the total equity, an exception will be
20 issued. The exception amount will be the amount
that will bring the total equity to the minimum
equity requirement amount. This exception can be
issued any time after total equity has been
calculated. Other exceptions are described in
25 Appendix IX.

The notice authorization module 950 will generate a
notice authorization table 975 from the exception
table 970. The notice authorization table 975 will
be rebuilt from the new exception table 970 at the
30 end of the batch process every night. The table
once built is forwarded to an outside source 976
for 24 hour around printing and delivery. The
notice authorization table 975 consists of the
following four notice types:

-130-

1) Sell-Out Notice - This
notice is generated by past due
Fed Call, or past due Cash Due
exceptions that have not yet
5   been reduced two business days
before the due date of the
exception;

2) Regulation T Margin Call
Notice - This notice is
10   generated by new Fed Calls. The
money amount for this notice
will only include the Fed Call
amount of the trades with
today's trade date;

15   3) Buy-In Notice - This notice
is generated by the Tech Short
(see Appendix IX) exception on
the thirteenth business day from
trade date;

20   4) House Call Notice - This
notice is generated by new House
Call exceptions.

At the end of the batch cycle, after the exception
table 970 has been updated, the old notice
25 authorization table 975 will be deleted. The
notice authorization module 950 will calculate the
due dates for exceptions that may appear on the
notice authorization table 975. The notice
authorization module 950 will add the authorized
30 notices to the notice authorization table 975 based
on exceptions on the exception table 970. The
notice due date is used to notify the customer when

the cash or securities must be delivered to the
securities trading firm. The notice display date
is used to control when a notice is to be
displayed. And in addition, the notice

5   authorization module 950 checks the NYSE extension
table 977 before determining that a notice is to be
displayed.

After all relevant exceptions have their notice due
dates and notice display dates are calculated, the

10   notice authorization module 950 writes to the
notice authorization table 975 all exceptions
having a notice display date equal to next business
date. This table contains all the notices that
need to be sent out for the next business date.

15   The notice authorization table 975 can then be
downloaded or transferred to a notice printing
service 978 for 24-hour notice mailing.

All active exceptions are placed on the exception
table 970. An exception will either be issued new

20   or aged, depending on the exception. Exceptions
that are issued new are removed from the activity
table 930 at the end of the next day. If the
condition exists again it will be issued as a
separate occurrence of the exception. Aged

25   exceptions remain on the exception table 970 until
the exception condition no longer exists.

The debit/credit interest module 954 sums and
multiplies the applicable margin interest rates to
derive margin interest. Debit and credit interest

30   will accrue nightly and be posted to the general
ledger 72 and the cash management system 54 at the
end of the monthly interest period.

An interest rate table 964 is accessed to determine
the interest rate to be used. Separate rates are
maintained for debit interest and credit interest.
An interest rate consists of a base rate and a
5   possible additional basis point offset rate that is
added to or subtracted from the base rate. The
offset rate be for either the range of customers or
the specific customer. The customer account
reference table 208, described above, indicates if
10   the customer has this preferential interest rate.
The range can be institutional, retail or employee.
If a customer has a preferential rate then that
offset will take precedence over the range offset
rate.

15   Debit interest is only calculated if there is a
debit balance in the margin account. The
customer's debit/credit balance is calculated by
netting the actual settlement date balances across
the margin cash sub-accounts 967. The market value
20   of technical shorts and illogical memo positions
(where P&B memo positions do not match the customer
settlement date positions, See Appendix X) are
backed out of the customer balances. This will
either decrease the credit balance or increase the
25   debit balance. After calculating the interest
rate, the daily accrued amount is added to an
interest rate detail table 972, where it is
maintained through the interest period. The
interest accrual is stored and reported by interest
30   rate periods. These periods change when the base
interest rate is changed and at month end.

A reporting payment and posting module 976 creates
customer statements and other industry required
compliance reports. The payment and position

function of the module 976 is to initiate the CAMS
payment requires update module 939 to make payment
request for customer accounts that are profiled (in
customer status 969) to be paid monthly. In a
5   batch process, the CAMS payment request update
module 939 will send an update to the CAMS module
54 (directing it to send a check) and update the
customer balance files 965, 967.

6.   Trade Clearance Module

10   The clearance module 52 depicted in FIG 3 of the
present invention tracks and records all clearance
activity related to a trade and in addition updates
the position and balance location files. As the
transaction processor of the present invention is
15   implemented for a securities trading firm, tracking
and recording clearance information includes
tracking security buy and sell transactions from
the time the transaction is booked until the time
that either 1) securities or payment is exchanged
20   in satisfaction of the trade on settlement date or
2) a settlement is received for failed trades. The
primary focus of the clearance module is the flow
of securities to and from accounts held by a
trading firm at various security clearinghouse
25   locations. That focus is on "streetside" receipt
and delivery, not customer ownership and control.
Thus, the clearance system output is integrated
into position and balance database 600 to update
the P&B location files.

30   Because the clearance module handles the receipt
and delivery of cash in satisfaction of trades, its
output is also an input for the cash management
module (CAMS) 54. As is be described above, CAMS

is the systems central management module for all
cash flow that relates to trades. As the clearance
module receives information on cash received for
securities delivered, the clearance module passes

5   that data to CAMS.

In addition to handling security receipt and deli-
very, the clearance module is also arranged to
provide reports of pending settlement activity, and
project pending settlement totals. Before

10 settlement date, the clearance module attempts to
consolidate clearing activity by pairing offset
trades. If a trade fails to settle on the pro-
jected date, the clearance module tracks the failed
trade until its ultimate settlement.

15 As the clearance module is implemented in the
three-tiered system, described in FIG 1, many of
the clearance module program elements to be
described below reside in the minicomputer
environment. Function modules related to position

20 and balance maintenance reside in the mainframe
environment.

i.   Trade is Booked

FIG 22a depicts the on-line process flow for
the clearance module after a trade is "booked" by

25 the trade entry module depicted in FIG 22a at 980.
The trade entry module 980 notifies several
modules, e.g. firm inventory, CAMS and the
clearance module of the trade booking.
Notification travels through the generic

30 distribution mechanism module as depicted at 982
and it arrives at a clearance system message queue
984. A receipt module 986 calls the a trade entry

accessor module 988 to gather clearance related
trade facts from the executed trade file depicted
at 990. Appendix VIII shows the data retrieved by
the accessor module 988. The receipt module 986
5   writes the data to a log file 992 and notifies a
clearance manager module 994 by sending a message
to its notification queue 996.

Asynchronously, the clearance manager module 994
awakes and reads the intermediate log file 992 for
10   the new trade details. To gather additional
clearance related details, the clearance manager
module 994 accesses one or more product
classification Systems (PCS) trees, 998, 1000,
using the product classification system access
15   routines 1002. One depicted PCS tree 998 maintains
the product class identifier for the security type
entered. The second depicted tree 1000 maintains
an attribute identifier number and the class name
of the security. Using location information
20   gathered with the PCS trees 998, 1000, the
clearance manager module 994 accesses information
from the product master database 1004 using a
product master accessor module 1006. Trade related
details furnish information as to the current
25   location of the securities, delivery instructions,
product and account access codes, and other
clearance related information.

The clearance manager module 994 next stores the
executed trade data sent from the trade entry
30   module 980. The clearance manager module 994
accesses a clearance IO manager 1008 to record the
expectation of the receipt or delivery of
securities to a clearance main file 1010 and a
clearance detail file 1012. The clearance main

-136-

file 1010 is the central repository of information
regarding the receipt and delivery of securities
and cash resulting from clearance activities. A
record is written to the clearance main file 1010
5   for each new trade. There is a one-to-one
correspondence between a clearance file record and
the original trade input data.

Associated with the clearance file is the clearance
detail file 1012. For every clearance activity
10 performed, a record is written to the clearance
detail file 1012. There may be one or more clear-
ance detail records for every clearance main file
1010 record. Activities which result in the
creation of a clearance detail records include: 1)
15 the original booking of the trade; 2) a trade
cancellation or correction; 3) a transmission of a
trade to a clearing entity; 4) acknowledgement of
clearance from a clearing entity; 5) a pair-off
confirmation; 6) receipt or delivery of full or
20 partial amount of the security quantity or trade
proceeds; and 7) trades going into fail status.
The clearance main file 1010 records and the
clearance detail file 1012 records are also sent to
the store and forward log, depicted in FIG 22a at
25 1014 to be shipped to clearance massive storage
tables for data in the mainframe environment.

The clearance manager module 994 also updates the
position and balance database 600 by invoking the
mainframe-based position and balance manager
30 routines depicted in FIG 22a at 1016. As described
above, the position and balance manager will create
appropriate updates to the trade date location and
other files in the P&B database 600.

Finally, the clearance manager module 994 invokes a
line-up manager routine 1018 to update line-up
projection file 1020, if necessary. Line-ups is a
function of the clearance module that reports

5   upcoming settling activity and projects end-of-day
settlement totals. The primary clearance window
time for the line-up file begin one day prior to
the settlement date and ends on settlement date.
Should a newly entered transaction expect to settle

10 within the time window of the line-up manager
module 1018, the transaction will be added
immediately to a line up file 1020. In addition,
the line-up manager 1018 sends a corresponding
entry to a mainframe-resident massive storage table

15 via the store and forward log 1014. Profit
information, displayed with the line up file 1020
is calculated asynchronously, using one or more
firm pricing modules 1022, 1024 that access the
product master database depicted in FIG 22a at

20 1006. Using the clearance IO manager module 1008,
the line-up manager module 1018 updates the line-up
file 1014 with price information and updates the
store and forward log 1014.

ii. Line Up Processing

25 Line-up processing is a function of the present
invention that provides the ability to report
pending activity and projected settled positions.
The repository of settlement information, the
"line-up file" is created largely in batch process

30 during the end-of-day processing. However, as
described above the line-up file can be updated by
immediately pending trades, on-line, during the
day.

-138-

As created in the overnight process, a line-up
module sorts through the position and balance
database 600 and extracts all securities that are
due to settle the next day. That list includes all
5   failed trades. In addition, the line up projection
file lists all files that are due to clear within
the near future, for example within the next five
days.

And on-line function permits users to view the
10 projected settlements. For each security issue, a
projection is created for the end-of-day versus
segregated positions. This involves netting
start-of-day positions with anticipated settlement
activity. The projections are used to 1) identify
15 short positions that show whether a borrow or
resale should be initiated, 2) identify surplus
inventory to be used in repurchase agreements or
loans, 3) pass cash projection figures to
facilitate cash management.

20   iii. Trade Pair-Offs

Another function of the clearance module depicted
in FIG 3 is its ability to pair-off trades. The
pairing-off function allows users to match all
offsetting buy and sell trades to reduce the
25 overall movement of cash and securities. The
function reduces physical movement of cash and
securities by offsetting buys and sells for the
same customer. Pair-offs can minimize trade fails
and reduce fail costs, because pairing-off reduces
30 the number of transactions that need to clear.

Pair-offs is a front-end function of the clearance
module. It's process flow is described in FIG 22b.

The function is invoked by user input from a PC
front-end module 1030. A minicomputer-based pair-
off manager module 1032, begins sorting through the
clearance main file 1010, identifying potential
5   pair-offs for a customer. As implemented in the
present invention, the pair-off manager module 1032
applies a sort criteria to determine potential
pair-offs by: settlement date, security type;
customer account; and by offsetting sell against
10 buys.

Once trades are paired, record of the pairing is
written to several files. The clearance manager
module 994 updates each trade record in the
clearance main file 1010, and in addition creates a
15 new pair-off entry for the clearance main file
1010. The clearance manager module 994 also
creates new entries in the clearance detail file
1012. The updates are completed using the
clearance IO manager module 1008. Corresponding
20 updates are made in the store and forward log 1014.

The pair-off manager module 1032 also uses the
clearance IO manager module 1008 to update several
pair-off record keeping files. A pair-off main
file 1034 contains one record per pair-off
25 regardless of how many trades are involved in the
pairing. The pair-off manager module 1032 assigns
a pair-off number to each record added to that
file. A pair-off activity file 1036 contains
records detailing subsequent events that occur to
30 the pair-off. A one to many relationship exists
between entries in the pair off activity file 1036
and entries in the pair-off main file 1034. As a
pair-off event occurs (e.g. payment made on a
pair-off) a record is added to the pair-off

-140-

activity file 1036. A pair-off trade relate file
1038 maintains the relationship between the
assigned pair-off number and the trades that the
pair-off relates to.

5   Finally, the pair-off manager module 1032 sends
notification of the pair-off to the cash management
module depicted in FIG 22b at 1040. The CAMS
module as described above uses pair-off information
to alter the payables and receivables it expects to
10 receive.

iv.   Receipt and Delivery

The process flow for clearing trades is depicted in
FIG 22c. The receipt and delivery process begins
in the evening or the morning before of trade
15 settlement. A background processing routine called
communication manager module 1042 sends lists of
trades that the firm expects to one or more
clearinghouse banks 1043 and communicates over wire
services with the clearinghouse banks while sending
20 the lists to them. Communication links for the
communication manager 1042 are created over
established wire services.

As an exemplary embodiment of the present
invention, it is recommended that the communication
25 manager module exist in a fault tolerant mini-
computer environment and communicate with the
outside world 1043 via transmission lines that
support a X.25 bisynchronous protocol.

The communications manager module 1042 establishes
30 CPU to CPU links with each clearing bank 1043 and,
after sifting through the clearance main file 1010,

sends to each clearing point bank a list of the
securities expected to clear there. An outgoing
communications log 1044 contains all records of
transmissions sent to the clearing banks 1043.

5   During the trading day, linked clearing house banks
     1043 transmit clearing information concerning both
     cash and security flows. An incoming
     communications log 1046 contains all records
     received from a clearing bank. A communications
10   control file 1048 contains a record representing
     the incoming and outgoing control information for
     each linked bank 1043. As the communications
     manager module 1042 receives information, it
     notifies the clearance manager module 994. Cleared
15   trade information can also be inputted on-line,
     using a PC-resident front-end function 1050.

     Asynchronously, the clearance manager module 994
     reads the contents of the incoming communications
     log 1046. For cash flow records, the clearance
20   manager module 994 sends the data to the CAMS log
     file 1040 for processing by the cash management-
     system module.

     For security movement records, the clearance
     manager module 994 first updates the clearance main
25   file 1010 and also creates a detail record for the
     clearance detail file 1012. The clearance IO
     manager module 1008 writes to those files and also
     updates the store and forward log depicted in FIG
     22c at 1014. For paired trades received, the
30   clearance manager module 994 invokes the pair-off
     manager module 1032 to make the proper file updates
     1032. Clearance manager module 994 also invokes
     the line-up manager 1018. In addition to updating

the projected line up file 1020, the line-up
manager 1018 also invokes the clearance IO manager
module 1008 to update the line-up location file
1056. That file contains netted security location
5   information on the basis of one record per bank and
product type.

In addition, clearance manager 994 invokes the
minicomputer-based position and balance manager
module depicted in FIG 22c at 1016. The P&B
10   manager module 1016 creates, updates and sends to
the P&B database 600 data that reflects the actual
receipt of securities. Those updates alter the
batch processed settlement date failed to receive
code "F/R" positions, described in the position and
15   balance section above.

v. Failed Trade Processing

When a trade does not clear by the end of the
business day of its scheduled settlement date, the
trade fails.

20   A trade can fail because full payment was not
received or the securities were not delivered on
the assigned clearance date. The terms "fail-to-
deliver" and "failed-to-receive" are used to refer
to open trades in the transaction processor that
25   have passed the assigned settlement date but have
yet to be fully cleared. If a trade has been fully
cleared for securities (i.e. securities are
delivered) but an open money difference remains,
the trade is still considered a failed trade. In
30   the securities trading industry, firms are subject
to regulatory reporting requirements for failed
trades. For certain security products, customers

and brokers must be notified of a failed trade at
specific time intervals.

FIG 22d depicts the process flow for gathering
failed trades. An overnight batch process, a fail
5   gathering module 1063, sorts through the clearance
main file 1010 copying failed trades to a fail file
1064. The fail file 1064 is the central file for
failed trades; it contains all fail costs and fail
ages. There is a one-to-one correspondence between
10   fail file record entries and actual failed trades.
A fail activity file 1066 contains activity details
associated with each fail. As fails are collected,
the batch fail processing module 1063 updates the
clearance main file 1010 and creates a detail
15   record for the clearance detail file 1012. Credits
are also computed for the cost of carrying the
securities and sent to the firm inventory module
depicted in FIG 22d at 1065 as described above.

FIG 22d also depicts the process flow as a failed
20   trade clears. As described above, the incoming
communication manager 1042 collects records of
incoming settled trades from clearing house banks
1060. A record describing the settlement of a
failed trade arrives in the incoming log file 1046
25   and is processed by the clearance manager module
994. As described above, the updates to the cash
management system module CAMS 1040, to the
clearance main and detail files 1010 and 1012, to
the position and balance manager module 1016 and to
30   the line-up manager module 1018 are completed.
However, in addition, clearance manager module 994
also invokes a fail manager module 1062. The fail
manager module updates the fail file 1064 and fail
detail file 1066. The fail manager 1062 also

-144-

computes a cost of carry credit and the cost of
fail for use by the firm inventory module 1068, as
described above.

7. Dividends, Interest and Redemptions
(DIR)

5

The present invention can include program modules
used to calculate and process all entitlements
related to a security position. As implemented for
a securities trading organization, the dividends,

10 interest and redemptions module (DIR) 70 collects
and records the disbursement of security
entitlements - dividends, interest and the
redemption of principal through amortization of
maturing securities.

15 As a matter of securities industry practice, the
DIR module 70 must track entitlement information
for securities products held in all firm, customer
and location accounts. Equity entitlements require
determining such facts as a declared ex-date,

20 record date, payment date, and dividend rate. Debt
entitlements require the derivation of such facts
as a record date, payment date, last accrual date,
interest rate, call rate and redemption rate.
Mortgage-backed debt products additionally require

25 a determination of factor date and factor rate.

Quantifying and allocating these entitlements is
based on who holds a position in the security in
relation to when its entitlement will be paid. The
account holdings are used to project record date

30 positions to facilitate distribution of cash or
stock dividends and to obtain record date
beneficial owners and locations for distribution

and collection of entitlements.

The DIR module 70 of the present invention provides
a centralized and automated system to identify
product entitlements and to process events related
5   to entitlement distribution. For example, changes
made to an entitlement cycle's payment dates or
changes in an account's holdings mandate changes to
the distribution amounts and collection of
entitlements.

10 In addition, trades that are in fail status over
record date or that are projected to settle within
an entitlement's interim accounting period, require
the distribution or collection of entitlements by
attaching a record keeping feature called "a due
15 bill" at time of clearance.

Entitlements in the form of money or stock must be
collected from the issuer of the product. The
ability to match receipts to receivables, age
outstanding receivables, issue claims for
20 receivables, and reconcile incoming receivables is
an important record keeping function of the DIR
module 70.

In addition, the DIR module 70 comprises a
"proofsheet adjustment" function to handle new
25 entitlement information as well as trade cancel or
correct information during the entitlement period.

As an integrated feature of the present invention,
the DIR module 70 uses information maintained by
other transaction processing modules and the
30 general reference databases 77, described above, to
maintain entitlement information. As entitlements

are received and distributed, the DIR module also
feeds information to other systems affected by DIR
activity.

FIG 23 depicts the high level process flow of the
5   DIR module 70.

The first step in the process is to organize basic
data on entitlements. An entitlement announcement
processing module (EASY) 1134 uses data from the
product master database 206 and outside financial
10   news services 1132 to create entitlement data
tables 1135 that list security products, normally
traded by the firm, on which entitlements become
due.

A start-up processing module 1139 begins the
15   entitlement distribution process by collecting
information on securities product, listed in the
EASY data table 135, whose entitlements cycle is
going on record date. Recording of record date is
important, because as a matter of securities
20   industry practice, all holders of securities on
record date will take the divident regardless of
whether they sell the securities before the actual
distribution date.

The start-up module 1139 takes information from the
25   EASY data tables 1135 on securities which are soon
to go on record date (1-3 days before record date)
and writes those records to a start-up file 1141.

From the time security products are listed on the
start-up file, until the actual record date, a
30   clean-up module 1244 continually checks the current
firm, customer and location holdings to report on

securities that are not positioned in such a way
that the firm can receive and distribute the
entitlement. The problem occurs when securities
are held in either the "BOX" or "TRANSFER" position
5 codes. The BOX position code means that the
securities are held in a name other than the
trading organization. Securities in "TRANSFER"
means that the securities are between holders. If,
on record date, either positions exist, the trading
10 firm will not be entitled to receive the
entitlement. From the time the security appears on
the start up file 1139, until its record date, the
clean up module 1244 checks entries in the start-up
table against firm holdings in the P&B tables 62
15 for "BOX" and "TRANSFER" positions held on
"started-up" products. A notification report is
distributed on each BOX or TRANSFER position until
record date.

For a given security on the start-up table 1141,
20 clean up processing stops on record date and
proofsheet processing begins. A proofsheet
processing module 1136, accesses the start-up table
1141, the entitlements data table 1135 and the
position and balance tables 62 to create balanced
25 listings of entitlements due by account. As in the
P&B tables 62, the netted firm and account
positions balance against location positions.
Proofsheet tables 1137 contain the balanced
entitlement data. Records of entitlement on the
30 proofsheet tables stay on those tables from record
date until the time of entitlement distribution.
In addition, the proofsheet module 1136 creates
proofsheet reports 1140 and implements part of the
phased distribution method.

-148-

Phased distribution is a standard practice in the
securities industry, where securities traders are
credited in firm accounts with entitlements on the
night before record date, while customers are
5   credited with entitlements on the night before
actual payment. Accordingly, as a part of
proofsheet processing, the proofsheet module 1132,
sends account-side entitlement information to the
firm inventory module 66, as described above.

10   A scheduling module 1142 looks again to the EASY
data tables 1135 to discover when entitlements are
going to come due for purposes of receipt and
distribution. For products with entitlements that
become due during the next day, the scheduling
15   module then checks the proofsheet tables 1137 for
positions in those products. The scheduling module
1142 creates a balanced entitlement transaction
table 1143 by account (firm, customer and
location), containing information on what
20   receivables and payables are due and owing. Notice
of the pending distribution is sent to the customer
accounts 68 and firm inventory modules 66 as the
second phase of phased distribution.

A receipt and distribution module (R&D) 1148 reads
25   the entitlement transaction tables 1143 and creates
updates for a payable and receivable table 1144 and
detail table 1146. For cash distributions, the R&D
module 1148 sends notification of the expected cash
movements to the CAMS module 54. Distribution of
30   security entitlements is maintained directly by DIR
70 through on-line input from sources like DTC 1154
and manual on-line input 1155. As CAMS receives
cash entitlements, it notifies the R&D module 1148
of the receipt, and R&D will update the receipt and

detail files 1144, 1146.

In addition, the present invention further
comprises a proofsheet adjustment module 1130 to
handle trade cancel and correct data and
5   entitlement alteration data.

i.   Entitlement Announcement
Processing

The purpose of the DIR entitlement announcement
processing module 1134 (EASY) is to determine and
10 record product periodic entitlement information.
EASY will use the static and external information
from the product master database 206 to formulate
periodic entitlement information and provide access
to this data based on entitlement dates. EASY
15 processing is done when a product is added, when
specific product changes are made, and at the
beginning of each new entitlement cycle.

Entitlement information comes to EASY from the
product master database 206 described above and
20 from external data sources 1132. Vendor services
are used to provide market information. For
example, a service known as "IDSI" provides
information on domestic corporate equity dividends.
The "J.J. Kenny" service supplies information on
25 municipal securities. The "Bond Buyer" service
provides factor information for mortgage backed
products sponsored by the federal government.

There is a considerable group of products for which
information is not explicitly provided, including
30 Corporate Debt, Treasuries (Bills, Notes, Bonds and
Zero Coupons), Asset Backed Securities, Money

Market Securities (e.g., CD's), Hybrid Securities,
(CMO's, STRIP's, etc.) and foreign debt and equity
securities. For those securities, entitlement
information is derived from the product master
5   database 206. Information in prospectuses and
indentures is entered on the product master
database 206. That information is considered
static, having little change throughout the life of
the product. It is used to derive periodic
10 entitlement dates and rates. Information provided
by external sources 1132 (vendor services) provide
dynamic entitlement data.

The creation of the entitlement list files 1135 is
shown in FIG 23a. Raw entitlement data comes from
15 updates to the minicomputer-based environment 32
product master reference database 206. (Data from
external reporting services 1132 is received in the
transaction processor system by the firm pricing
modules 218 described above (See FIG 7)). The raw
20 data together with product master database updates
are sent to the store and forward log 43 for
transport by the store and forward modules 41, 42
to a mainframe-based general storage table 44. The
store and forward notification module 46 sends a
25 message that information potentially relevant to
DIR has been received.

Notification is passed (using a queue 1160) to an
EASY program initiation module 1172.
Asynchronously, the program intiation module 1172
30 interprets messages received from the notifier 46.
The product data received are classified by
intiation module 1172 to determine if further DIR
processing needs to be invoked. If entitlement
will be affected by the product information, the

initiation module 1172 next determines the
criticality of the update and accordingly sends the
data for either an asynchronous or background
processing. Batch records have a low priority and
5   will be processed in background, but critical
events will be processed on-line. Update records
for batch processing are stored in a batch
processing log 1171. Critical records are fed to a
queue 1176.

10 A subsequent action determination module 1178
receives data from the queue 1176 and sends the
data to a next product recognition process -- add
1182, modify 1190, delete 1192, etc. -- based on
the type of transaction code that came with the
15 data from product master 206. To determine the
subsequent entitlement events, the subsequent
action module 1178 uses references maintained in an
event reference table 1180 that are keyed by the
transaction code.

20 The entitlement add, modify and delete modules
1182, 1190, 1192 derive entitlement dates, amounts
based on product-type information. The subsequent
action module 1178 looks at the new product data to
determine whether entitlement cycles should be
25 built, modified or deleted and invokes
corresponding nodules.

For building new entitlement cycles, the following
procedure is used, based on product information
contained in the input as sorted against entries in
30 an entitlement type table 1181: If a maturity date
is not provided, the add module 1182 will then call
a build maturity cycle module 1194 to determine and
build the cycle.

If a "call" feature needs to be created, a call
build call cycle module 1196 is invoked determine
and build the cycle.

5   If the dividend rule feature indicator field is
"on", in the input record, then the add module 1182
calls a build anticipated dividend cycle module
1198 to determine and build the cycle.

If the product feature dividend pay indicator field
is "on" in the input record, the add module 1182
10   calls a build announced dividend cycle module 1200
to determine and build the dividend cycle.

If the interest feature indicator field is "on" in
the input record, the action module 1182 calls a
build interest cycle module 1202 to determine and
15   build the interest cycle, if necessary.

If the factor feature indicator field is "on" in
the input record, the action module calls a build
principal paydown cycle module 1204 to determine
and build the cycle, if necessary.

20   For entitlement on products that cannot be derived,
default entitlement dates are supplied on a default
entitlement table 1206.

The entitlement add module 1182 next writes the
cycle information to the entitlement lists 1135.
25   Tables exist for bond maturity entitlements 1208,
calls 1210, factor principal paydowns 1212,
interest 1214, dividends 1216 and dividend
alternatives 1218. Related to those tables are two
common tables: an entitlement group table 1220 and
30   an entitlement header table 1222.

With the tables updated the subsequent action
module 1178 calls the notifier module 1186 to
announce to other DIR modules of the table changes
if the change was critical: scheduling, start-up,
5   and proofsheet processing. The modification module
1190 and the delete module 1192 perform in a manner
similar to the add module 1182.

ii. Start-up, Clean-up Processing and Phased Distributions

10   FIG 236 depicts start-up and clean-up processing
and phased distribution.

For stock and cash dividend entitlements, the ex-
dividend date from the entitlement table 1135 is
put on the start up file 1141 in the firm-payment
15   date field. For stock and cash dividend
entitlements, the start-up scheduling module 1139
will read the firm account settlement date P&B
table 604 for any settled firm positions in the
product. If no position is found, the start-up
20   scheduling module 1139 will also read executed
trade file records 1240 for form account trades
with a projected settlement date less than the ex-
date or record date depending on the product type.
If either is found then the modules will set a
25   firm-position flag on.

A batch report process routine 1242 reads the
updated start-up schedule table and creates a
written report 1140 listing products that are
beginning an entitlement cycle. The start-up
30   schedule file 1141 is read for products with a
start-up flag set to 'Y' and a start-up date equal
to the current day. The reports 1140 will note the
type of entitlement, product number, description,

record date, payment date, first and last accrual
dates, ex-dividend date, interim dates, rate and
whether a product has clean-up position. The
reports 1140 have a different format for each
5 entitlement type.

As discussed above, start-up and clean-up
processing are two totally distinct but related
steps in entitlement processing.

An entitlement cycle is "started-up" normally 3
10 days before the record date or on the date one day
preceding the ex-divident date of a security. For
securities with stock and cash dividends, as a
matter of security record-keeping practice, start-
up occurs on a security's ex-dividend date. As
15 part of the phased distribution scheme, firm
accounts are paid all ex-date entitlements based on
the projected settled record date positions held.
As a part of a start-up processing, a firm ex-
dividend payment report is created. This payment
20 sheet is adjusted for activity between ex-date and
record date by proofsheet adjustment functions
described below. In a batch process, the start-up
module reads the EASY entitlement data tables 1135
to locate products that have not. The process flow
25 for the start-up scheduling subsystem is depicted
in FIG 23b. The processing to create the
notification table 1141 is performed in background.
On an end of business day basis, the entitlement
lists 1135, described above, are read to locate
30 products that have not already been started-up and
whose start-up date is equal to or less than today
(if today is followed by a weekend or holiday these
days are included).

For the products selected, a start-up scheduling
module 1234 first creates a new entry in the start
up file 1141, and sets the start-up and clean-up
flags and start-up and clean-up dates. The start-
5   up file marks the data when the security was placed
on the list. The clean-up flag denotes how long
the clean-up process (described in detail below)
will be performed on that particular entry.

The clean-up process, distinct from start-up, uses
10   the start-up file 1141 to locate any security
positions in a BOX or TRANSFER positions, whose
products are on the start-up table 1141. Thus,
where start-up's function is to create the initial
start-up table, it is the function of the clean-up
15   module to continually check from initial start-up
until record date, whether there is a BOX or
TRANSFER position for that security.

Using the products listed on the start-up table
1141, the clean-up module 1244 reads the P&B
20   location files 606,612 to locate TRANSFER and BOX
positions in those securities. Once those
positions are collected at the end of each day, the
clean-up module 1244 creates a clean-up report
1140.

25   A phased distribution module 1250 allocates stock
and cash dividends on an ex-dividend record date
basis to firm accounts. Phased distribution means
processing of the firm entitlements at phase in the
entitlement cycle, earlier than the customer
30   allocation of entitlements to customer accounts.
Phased distribution is a general record-keeping
procedure used by securities trading firms.

-156-

The phased distribution module 1250 normally begins
processing of firm account entitlements on the
night before the ex-dividend date of the dividend
entitlement. However, the module can be

5   synchronously invoked during the trading day, when
there has been a time-critical change to the start
up file 1141.

As mentioned above, the start-up scheduling module
has marked potential securities for entitlement

10   due. The firm-payment date field for all
applicable securities in the start up file 1141 has
been given its ex-date. The phased distribution
module 1250 searches the start-up file 1141 for
records where the ex-date and firm payment date

15   equals the current date. For all the records where
ex-date and firm payment date equals the current
date, the phased distribution module 1250 checks
the P&B firm settlement date table 604 to determine
if a position is held. Where there is a product

20   match, the phased distribution module 1250 creates
payable or receivable records to be processed by
the DIR entitlement distribution module
(scheduling) 1142 and the general ledger interface
module 72. The updates are sent to the entitlement

25   distribution module (scheduling) 1142 and the
general ledger interface module 1254 using the
generic distribution mechanism 1252 described
above. For all entitlements, the phased
distribution module 1250 creates record information

30   and notifies the firm inventory module 66. As
described above, the firm inventory module 66 will
invoke the P&B manager module 64 to create P&B
table updates. The phased distribution module 1250
also outputs to the reports on the firm ex-date

35   entitlements created.

iii. Proofsheet Processing

Proofsheet processing module 1136 creates and
reports on security entitlements due, using the
entitlement cycle information created by the EASY
5   1134 and start-up modules 1139, and firm position
data located in the P&B 62 and other files.
Proofsheet processing functions enable users to
gather pending entitlement information for
securities actually held, such as firm position on
10 a stock's ex-date; record date positions; call date
positions (lottery results if the call is partial)
and "pre-record date" positions for the securities
whose record dates precede payment date by one day.

The process of creating proofsheets requires
15 matching the general entitlement cycle data created
by EASY 1134 with the actual position data
maintained in the position and balance files 62.
For example, record date proofsheets are created in
daily batch processing by collecting the EASy
20 record date entitlement information and matching it
with position and balance file settlement positions
and trade information located in the executed trade
and other files. On a security's ex-date, the
proofsheet processing modules captures position and
25 balance trade-date positions 604, 602, 606 and
projects the settled record date position
entitlements for those pending trades.

Proofsheet data created for record date settled
positions is used to create entitlement payables
30 and receivables, as will be described below.
Because those receipt and deliver functions require
up-to-the-minute entitlement information, the
proofsheet data can be updated by a proofsheet

adjustment and override functions. Override and
proofsheet adjustment are also described full below
in FIG 23d.

An overview for creating a proofsheet as of record
5   date is shown in FIG 23c. Record date proofsheets
are created at the end of a business day for the
next day, during batch cycle processing.

An entitlement information gathering module 1270
begins reading the EASY entitlement information
10   files 1135 described above searching for all EASY
records with a record date of the next day. Those
records are written to an intermediate log file
1274. A proofsheet matching module 1276 reads the
intermediate log file 1274 and attempts to match
15   those records against the P&B database settlement
date position files 610, 608, 612. The matching
module 1276 will exclude all position and balance
matches when the securities are either in a (box),
safekeeping (SFK), or transfer (TRANSF) position.
20   As a matter of securities recordkeeping practice,
securities in safekeeping, box and transfer will
not render any entitlements for a firm to process.
All entitlements for those security positions go to
the customer -- there is nothing that a firm needs
25   track.

The matching between the position and balance
database account positions and the EASY entitlement
information forms the basis of the record date
proofsheet. Records on the matches are written to
30   a proofsheet file 1282.

For certain stock positions such as fails, stock
borrows, stock loans, and repos, the transaction

processor of the present invention maintains detail
records. The details are contained in files such
as the clearance fail activity file depicted in FIG
23c at 1066. The transaction processor
5   incorporates those details into its proofsheets,
making them very useful to users. A detail
collection module 1284 asynchronously access the
clearance module fail 1064 and fail activity 1066
files, and the position and balance memo file 614
10   for "repro", and "stock borrow" and "loan"
positions. The gathered details are written to a
proofsheet detail file 1289. When proofsheet
reports are generated, the proofsheet database
details are included.

15   Another aspect of the proofsheet processing process
is the creation of due bills. As a matter of
securities practice, entitlements accruing to
trades that are in a fail status on record date
cannot be given to the beneficial owner. On record
20   date, the owner is the original security holder --
any entitlement due will go to the original owner,
regardless of the fact that the trade will
subsequently clear, albeit after the original
settlement date. To permit the beneficial owner to
25   receive the entitlement that it would have received
had the trade settled as planned, security firms
will as a matter of practice, send a due bill along
with the trade when it finally settles. Thus,
certain trades require a "due bill" upon outgoing
30   delivery and certain purchase trades require a due
bill receipt acknowledgment upon incoming delivery.
The due bill module 1290 creates a file 1298 to
track the due bill entitlements and insure that as
a stock clears the due bill is sent or received.

The trade entry modules 48, described above, sets a
due bill indicator on the securities that require a
due bill should the trade fall. The due bill
module 1290 accesses the proofsheet table 1282,
5   searching for trades whose failed status will
require a due bill.

For each pending trade that needs due bill proces-
sing, other program modules are invoked to
calculate the amount of the due bill 1292 and
10   determine the due bill type 1294.

A due bill calculation module 1292 is invoked to
obtain rate and other information from the EASy
entitlement database files 1272. In order to
calculate a due bill, different rates are used,
15   depending upon the entitlement type.

The determine due bill type module 1294 is invoked
to determine the type of due bill required,
promissory note or due bill check, for each
associated trade. The module makes that
20   determination by comparing the projected settlement
date of the trade with ' a payment date of the
entitlement. The due b..l module 1290 updates the
proofsheet file 1282 with the results of the
calculation. The module 1290 also updates the
25   detail file 1289 and to a due bill file 1298. Due
bill amounts are also sent to the clearance module
52.

A create proofsheet module 1295 read the data
collected in the proofsheet table 1282, the due
30   bill file 1298 and the proofsheet detail file 1289
and combines them with the proofsheet intermediate
file 1282 to create a proofsheet report 1296. The

-161-

create proofsheet module 1295 sums the proofsheet
report records in a proofsheet header file 1297.
Corresponding details remains in the proofsheet
detail file 1289.

5   iv.   Proofsheet Adjustment

From record date to the date of entitlement
payment, and even beyond the payment date, changes
can occur to the firm's position holdings that
affect the entitlement schedule. For example, a
10 trade could be cancelled or modified. Stock
settlements can fail across a record date.
Entitlement payment schedules can change. Stock
dividends can be rescheduled, cancelled, or
increased. A company can declare a spurious stock
15 split. The proofsheet module of the present
invention maintains accurate and up to the minute
proofsheet information by correcting the record
date proofsheet with update information. The
process flow for the proofsheet adjustment
20 procedure is depicted in FIG 23d.

Trade and settlement activity 1300 is sent from the
minicomputer environment 34 to the mainframe
environment 36 through the store and forward
module 42, as described above. On the mainframe
25 side, the store and forward module 41 copies the
data into an adjustment process storage table 1306
and the store and forward notification module 46
triggers the subsequent processing by writing a
message into a queue 1307. Changes in trade date,
30 trade quantity, process date, projected settlement
date, clearance date, clearance quantity, due bill
indicator and amount, will be part of the data
provided.

An adjustment driver module 1308 is triggered by
the notification module message to process all
trade and non-trade adjustment information. Once
triggered, the adjustment driver module 1308

5   invokes a pre-processing module 1310 to read all
the transaction records in the adjustment process
table 1306. The pre-processing module 1310
attempts to indicate off-setting transactions to
eliminate record breaks in later adjustment

10   processing. For the records to be processed, the
pre-processing module 1310 returns control to the
driver 1308 with information concerning the
adjustments needed.

Update information can mandate one of four general
15   types of updating. Trade actions, such as cancels
and corrects or failures can mandate adjustment to
current entitlement payment (distribution), if the
entitlement payment date has not occurred, and
adjustment to post distribution cycles if

20   entitlement payment has occurred. Product master
database updates or late breaking financial news
can necessitate the need to make immediate changes
to the product entitlement cycle schedule, and
subsequent updates to the proofsheet balances.

25   The driving routine will invoke one or more
asynchronous processing modules 1312, 1314 and 1316
to handle each adjustment situation.

A current proofsheet cycle adjustment module 1312
is invoked by the driver module 1308 after it has
30   determined that the cycle status is in a pre-
distribution phase and that the change affects a
firm, customer or location position prior to
payment. For each adjustment record, the current

proofsheet adjustment module 1312 is involved to
either recreate the entire proofsheet or update a
part of it.

A proofsheet recreation module 1320 extracts new
5   position and balance positions from the position
and balance database 62, applies all trade and
non-trade activity to updates the position, update
the cycle status and creates an adjustment
transaction to update the proofsheet table 1282.
10 An entitlement record date change is an example of
a charge that would require recreation of an entire
proofsheet. Record of proofsheet adjustment is
maintained in a proofsheet adjustment file 1323.

An adjust proofsheet module 1324 handles changes
15 that do not require major changes to the
proofsheet. Single trade cancel and correct
information such as the correction to the amount of
securities purchased is an example of a change that
would affect only a position on the proofsheet.
20 The update proofsheet module 1324 would change and
create an adjustment transaction to update the
proofsheet database file and the proofsheet detail
file. Record of the adjustment is also kept in a
proofsheet adjustment file 1323.

25 The adjustment driver module 1308 invokes a past-
cycle adjustment module 1314 to make trade changes,
based on adjustments to past proofsheet data in the
proofsheet file 1282. The past-cycle adjustment
module 1314 will create a new proofsheet file
30 entries 1282, 1289 as needed, create an updated
entry for an existing position, invoke the due bill
module 1290 to create a new promissory note or
check and update the proof adjustment file 1323.

-164-

Entitlement changes received from the product
master database will sometimes require proofsheet
adjustment. When a product entitlement update is
encountered, a pre-processing module 1326 to
5   determine the criticalness of the change, according
to some specified criteria. The preprocessing
module 1326 will first determine the type of change
that has been made. For example, an adjustment may
be required as a result of a late dividend
10 announcement during the days' trading. A product
classification tree 1328 is used to categorize
product type charge by importance.

If there is no immediate need to make the product
master update, the criticalness module 1326 sends
15 the record to the batch 1171 file for overnight
processing by the entitlement announcement
processing module 1134 as described above.
However, if a critical update is required, a EASY
file update module 1338 will first correct the EASY
20 files 1135 and invoke the current proofsheet
adjustment modules 1312 or the post-distribution
proofsheet modules 1314, as needed.

v. Distribution

The distribution module 1142 of DIR makes
25 allocations of the entitlements to customer and
other accounts within the firm. Distributions are
in the form of trader and customer position
adjustments, checks, wire payments and stock
payments. Distributions are made on a specific
30 date, as determined by the type of entitlement and
the beneficial owner.

The process flow for the distribution module 1142

is shown in FIG 23e. A distribution module 1340
creates the distribution records and store them in
an entitlement history table 1346. The
distribution records are based on information from
5 the schedule entitlement transaction tables 1143,
the proofsheet header file 1297 the proofsheet
detail file 1289 and the proofsheet schedule file
1282. The distribution records will carry the
customer and account name, the entitlement-type,
10 the amount of money or stock due, and the projected
effective date of payment. In most cases, the
effective payment date will be the actual payment
date. However, the transaction processor of the
present invention permits phased distribution of
15 entitlements. As a matter of securities practice,
traders may be credited with entitlement
distributions on dates different from customers or
security beneficiaries. The distribution module
1340 will create two separate distribution records
20 for the entitlement history table 1346.

The entitlement history table 1346 contains a
history of all entitlement distribution
information. However, updates to the file are
first written to a one-day table 1348, containing
25 all distribution records that were collected during
the previous night's batch processing. A DIR
notification module 1349 feeds the one-day table's
1398 contents to other transaction processing
modules, the firm inventory 66, P&B 64 and customer
30 accounts 68 modules depicted as 1343 and 1344.

vi. Receipt and Reconciliation

The DIR receipt and reconciliation module 1148 man-
ages the movement of all entitlement monies and

-166-

securities. The DIR receipt and reconciliation
module 1148 interfaces with the cash management
system module 54, the position and balance module
64 firm inventory module 66 and customer accounts
5   module 68 to process entitlement cash flow to
handle entitlement security movement. Thus, the
movement of entitlements is integrated with the
main cash and security flow modules of the present
invention.

10 There are two basic functions to the receipt and
distribution module 1148 of the present invention.
The first function is to set up payable and
receivable records for the entitlements. The
second function is to record the receipt or
15 remittance of entitlements.

The process flow for creating payable and receiv-
able entitlements (and then tracing their
remittance) is depicted in FIG 23f. Using the
records from the schedule entitlement history table
20 1346 and the one day table 1348, a receipt and
reconciliation processing module 1350 creates
payable and receivable records for the entitlements
listed. The records are written to a DIR payable
and receivable file 1352.

25 Payment and receivables are maintained for firm and
customer accounts and streetside location accounts
in a manner similar to the account and location
files are maintained in the P&B tables 62.
Streetside and accounts-side payable and receivable
30 entitlement records must offset. A link
entitlement file 1354 contains cross-references
from the entries in the payable and receivable file
1362 with entries in the proofsheet detail file

1289.

Copies of all DIR payable and receivable file
records 1362 pertaining to cash entitlements are
sent to the cash management system module 54, for

5   integration into the CAMS system-wide integrated
payable and receivable file 56 (see FIG. 3)
described above.

All DIR payable and receivable records for
quantities of security entitlements are sent to the

10 position and balance module 64. To make the
records available to P&B, the receipt and
reconciliation module 1350 sorts though the DIR
one-day table 1348 and feeds all security quantity
records to the position and balance settlement

15 files on the night before entitlement payment.

The receipt and reconciliation module 1350 performs
a number of reporting functions. A due bill
production module 1364 selects payables that
require a due bill and gives information to a

20 printing module for due bill printing. An external
data interface module 1362, notifies banks of the
need to present securities or coupons for interest
payment.

FIG 23f also shows the process flow for the receipt

25 of entitlements. As cash funds are received, the
CAMS module 1356 makes record keeping updates and
subsequently notifies the DIR module 70 of the
receipt. Typically, the payment is a lump sum
payment that must be allocated to the various

30 accounts on the entitlement history table 1346.
Initially, the DIR module 70 is notified via the
store and forward modules 41, 42 of the CAMS module

cash receipt. A matching module 1359 receives the
records and updates the DIR payable and receivable
files 1352.

The matching module 1359 then allocates the
5   entitlement receipts to the various parties by
cross-referencing entries in the payable and
receivable file 1352, with entries in the link
processing file 1354. The allocations of
entitlements to accounts can be based on an
10   appropriate formula. The updates are made by the
matching module 1359 to the distribution tables,
first to the one-day table 1348 then to the history
file 1346. The DIR notification module 1349
notifies other transaction processor modules firm
15   inventory 1343 and margins 1344 of the distribution
updates.

8. General Ledger Interface

The general ledger interface module (GLI) 72
depicted in FIG 3 is the accounting core of the
20   present invention. GLI's central function is to
systematically update the firm's general ledger
with all transaction activity. On an on-line,
real-time basis, GLI accepts the transaction events
and, at the end of the processing day for each
25   branch of a securities trading firm, automatically
creates accounting records that reflect transaction
activity. The records are netted together to
create the summary journals that update the general
ledger. A reconciliation process is designed to
30   assure that the processing updates to the
transaction processing modules are accurately
reflected in the general ledger. On-line inquiries
provide the ability to review control account

postings and "as of" postings. Further, depending
on posting date, GLI provides the ability to
selectively post "as of" prior period activity
processed in the current period, to the prior
5   general ledger accounting period.

The process flow of the GLI module 72 is depicted
in FIG 24. GLI has two process phases: on-line -
during the transaction day - and over-night batch
processing. During the day, the GLI module
10   collects reports of transaction events from other
systems and provides edits. In batch the GLI
module 72 classifies those records, creates
additional offsetting account entries, formats them
and creates GLI detail records. The detail records
15   form the basis for the journal entries to a firm's
general ledger.

The processing begins as the transaction processing
modules (e.g. 1370, 1372) update their files and
send those records to a GLI automatic journal
20   processing module 1374, 1376. The forwarded record
contains information needed to create GLI detail
postings to a GLI detail file 1410, including a
source module information and position information
(e.g., product number, trading account, and
25   inventory type).

The automatic journal processing modules 1374,
1376, existing both on the mainframe and the
minicomputer environments, function as a book-
keeping mechanism to take the raw data from the
30   source systems to create general ledger detail
postings. Automatic journal processing modules
1374, 1376 perform two basic functions using the
raw data: classify and uniquely identify the

incoming record. A second detail explosion module
creates the general ledger entries 1408.

The automatic journal processing modules 1374, 1376
perform classification functions by accessing data

5 from the general reference databases 76, 77 to
create ledger destination codes that will group
each transaction event entry in appropriate ledger
journal sub-entries. A product classification tree
1386, 1388 groups all the security products by

10 their journal entry destination. Product
classification modules 1379, 1377 traverse the
trees to obtain the journal entry destination for
each record. Further product information used to
create destination codes is found in the product

15 master general reference database (206 in 76, 77).
The trading account master database depicted (204
in 76, 77) enables the automatic journal processing
modules 1374, 1376 to translate an account number
into trading desk, office, and company journal

20 sub-heading. Input information contained on the
record helps make up the remainder of the general
ledger account record entry.

With the records classified and identified in a
format for further processing, the automatic

25 journal processing modules 1374, 1376 write the
records to an intermediate log file 1390. In
addition, a file 1392 is maintained keeping running
totals to project the end-of-day account balances
of the transactions as classified.

30 The mainframe-based automatic journal processing
module 1376 writes to the intermediate logs 1390,
1392 directly. The minicomputer-based automatic
journal process module 1374 outputs to the store

and forward log 43. The store and forward modules
41, 42, as described above, writes the data to a
massive storage table 1391. The store and forward
notification module 46 awakens a GLI mainframe

5   environment interface module 1395. To retrieve the
data from the massive storage table 1391 and add it
to the GLI intermediate log files 1390, 1392.

As a matter of general accounting practice,
accounting books are kept on a double entry basis,

10   (credit and debit) and book entries balance.
Accordingly, each transaction input sent to GLI
from a transaction processing module (such as trade
entry 48, DIR 70) will require at least two GLI
accounting entries. Some transaction input can

15   contain enough data to create the multiple GL
entries. The multiple entries are created from the
GLI classified intermediate records 1390, 1392 in
the detail explosion phase, described below.

Other transaction input do not contain enough

20   information to create an entry and offsetting
entry. These entries must either be created or the
transaction data must be listed to other
transaction data to create the link.

For transaction events that will not translate into

25   both the entry and offsetting entries, the
automatic journal processing module 1374 and the
GLI minicomputer interface module 1395 will trigger
a control account module 1394 to create the offset-
ting entries needed to accurately place the

30   transaction event information into the general
ledger.

Depending on the transaction event, an offset post-

ing could be either 1) a system to system control
account linking entry; 2) an intercurrency control
account entry; 3) a transaction processing system
to non system suspense account entry; or 4) a

5   manual suspense account entry.

Control accounts are general ledger accounts used
to establish accounting links between the flow of
information between two transaction processing
modules that independently pass GLI updates for the

10   same transaction. Transaction processing modules
create a general ledger postings that are comprise
only one side of the accounting entry. The control
account module 1394 creates control account
postings as the offsets to those otherwise one-

15   sided entries. For example, when securities are
sold, the trade-entry modules 48 will create an
accounting debit trade receivable journal entry,
and the control account module 1394 creates an
offsetting credit to the control account entry.

20   The firm inventory module 66 will initiate a credit
entry to inventory. GLI creates with the control
module 1394 another offsetting debit entry to the
control account causing the net effect of the
entries in the control account file 1396 to

25   balance. On the other hand, if both sides of an
accounting entry are created by the same
transaction process input, in the transaction
processing module, no control account record is
created.

30   Creation of intercurrency control account records
initiated by the control account module 1394 to
offset native, base, and consolidated currency
ledger entries. Within the individual ledgers,
each side of the intercurrency account is posted by

a different module. For example, the debit to an
inventory journal and an offsetting credit to an
intercurrency account journal is initiated by the
firm inventory system, while the credit to a trades
5   payable journal and an offsetting debit journal to
the intercurrency account is initiated by the
trade-entry module. Assuming that both systems
pass their side of the entry to the GLI module, the
intercurrency account within the base and
10 consolidated ledgers should net to zero. The base
and consolidated intercurrency account acts as a
control account to verify that both sides of an
inter-system data flow are passed and properly
posed by the GLI module.

15 When one side of an entry is initiated by a
transaction processing module and the other side of
the posting is a non-transaction activity, GLI
posts the transaction processor entry to a regular
general ledger account. Records are extracted from
20 a GLI detail file 1410 during the nightly batch
cycle to be described below. This file contains
the information required to update the non-
transaction account accordingly.

Mutual suspense accounts entries are created by the
25 control account module 1394 when a transaction
processing module cannot match an entry to a
payable or receivable by the end of the business
day. The feeding source transaction processing
module updates its balances and calls the GLI
30 automatic journal process module with a "special
business event" record. The automatic journal
process module 1376, will invoke the control
account module 1394, create an entry to update the
account which corresponds to the source system

balance, and then post the "unknown"
payable/receivable to a suspense account. For
example, if at the end of the processing day the
CAMS module is unable to match a dividend payment

5 to a corresponding receivable on the integrated
payables and receivable file, it posts an "unknown"
item on the IPR file. Simultaneously, DIR passes
the update to GLI with a specific business event
code and GLI creates a manual suspense account

10 posting to reflect the unknown receivable.

Overnight batch processing begins with an entry
reclassification function.. As described above,
during the processing day, the GLI module uses the
account cross-reference table 1378, 1380 the

15 product classification trees 1386, 1388 and the
trading account master database 204 to translate
the transaction processor source module data into
components of the general ledger account key.
Because those database files can be updated during

20 the day, general ledger classifications created for
the previous days may no longer reflect the current
classifications. To insure the integrity of the
general ledger, a GLI reclassification process
module 1398 updates any GLI intermediate posting of

25 control account file that should be affected by a
change in the database.

In batch mode, the reclassification process re-
builds the general ledger account records for the
projected source system balances, using the updated

30 current end-of-day account cross-reference table
and the updated centralized databases 44. The
reclassified balances are compared to the
originals. The re-classification module 1398 will
change the classification of any entry and if

necessary its affected control account records
1396. The reclassified projected closing balances
are written to a reclassified closing balance file
1402 and the reclassified GLI records are written
5   to a reclassified intermediate file 1404.

In batch mode, the reclassification module 1398
will also rebuild the general ledger account file
records for yesterday's transaction processor
module balances, using data stored in a yesterday's
10 balance file 1406. Every classification change to
those files will create a new transaction processor
module detail 1392 record to correct and create
offsetting entries for the mistake. The
reclassification module 1398 will update the
15 general ledger balance file by reversing the
balance out of the old general ledger account and
applying it to the new general ledger account.

A detail explosion module 1408 takes the now accu-
rately classified and identified source records and
20 populates them into as many journal entry details
and offsetting entries as necessary to properly
update the general ledger.

Explosion of the GLI intermediate file 1390, 1392
updates is accomplished by accessing account
25 cross-reference tables 1378, 1380. The account
cross-reference tables comprise table structures
that contain codes that will determine functions to
be performed on the input. Each source module
record is mapped to a three-digit general-ledger
30 code kept on the account cross-reference tables
1378, 1380. That code is used to clarify the
entries that will be created from the raw data to
appropriate journals on a firm's general ledger.

-176-

Associated on the table account cross-reference
tables 1378, 1380 with each three-digit general
ledger journal code is a corresponding three-digit
code for the offsetting journal entries which will
5   have to be created from the raw transaction data.
The automatic journal process module 1374, 1376
select the correct code based either on the type of
business event or the identification code of the
transaction processing module input.

10   The detail explosion module constructs a unique
reference key for each record, consisting of three
distinct parts, for each detail record. Using
source system supplied information, GLI detail
explosion module 1408 constructs the reference key
15   using the source system reference number, the
matching system's reference number, and a
sequential numerical suffix which is incremented by
one each time a detail posting is created. The
reference number is an important tool that allows
20   users to research transaction information and
accounting entries in GLI and back to the
transaction processing module supplied data. All
detail records are written to the GLI detail file
1410.

25   For all offsetting position in the control account
file 1396, a GLI control account matching module
1412 attempts to match control account entries.
The control account matching module 1412 totals all
control records in control account unmatched file
30   1396 for a transaction event. If the total of the
entries is zero, the control account entries are
considered "matched." In the GLI detail file 1410,
offsetting entries are marked. Matched control
records are also written to a control account

balance file 1414. However, when the control
account unmatched file 1396 does not net to zero,
the entries remain in unmatched status.

As detail file records are created, the detail

5 explosion module 1408 encounters "as of" records
that require special handling. "As of" entries are
entries that have arrived on the current day, but
whose effective date is set to an earlier date. As
a matter of accounting practice, "as of" records

10 can be posted in certain cases to either the
current accounting period when the "as of" was
received or the immediately prior period.

Three program modules facilitate "as of" proces-
sing: an "as of" adjustment module 1416, an "as

15 of" cancellation module 1418, and an "as of"
inquiry 1420 module.

"As of" input is passed to the GLI module as
transactions with "as of" dates. Any "as of"
transaction passed to the GLI module on the first

20 two business days of the current month's effective
date are automatically posted to accounting
journals for the current period and then set up as
"as of" reversal postings to the journals of the
previous month by the automatic journal module

25 1374, 1376. The detail explosion module 1408
stores the prior period adjustment on an "as of"
file 1422. The effect of the reversal within the
firm's general ledger is to reverse the postings
from the current period and apply them to the

30 immediately prior period. An "as of" inquiry
module 1420, provides on-line review of the
automatically prior period posted "as of"
adjustments.

Any "as of" transaction processed after the second
business day of the current month (notational "as
ofs") are automatically applied only to the current
period on the GLI detail file 1410. With the "as

5 of" inquiry module 1420, the user may review the
details of the notational "as of" posting on-line
and select those items which should be applied to
the prior period. An "as of" adjustment module
1416 then creates the appropriate detail postings

10 to reverse the item out of the current accounting
period and apply them to the immediately prior
period. The "as of" prior period reversal entries
are maintained on a GLI "as of" file 1422. An "as
of" cancellation module 1418 provides the facility

15 to cancel notational "as ofs" that were manually
posted to the prior period. The effect of the
cancellation process within the general ledger is
to reinstate the original "as of" posting to the
current period.

20 After the detail records have been classified,
identified and exploded, a summation process module
1440 reads the detail file 1410 and sums the
journal postings to create summary journals. The
summary journals are used to update the general

25 ledgers 1442. Non-trade related accounting details
are also summed by a separate module 1444 to create
a non-trade related posting file 1446. For audit
trail purposes, each summary posting to the general
ledger maintains a link to its supporting details

30 through reference key table (1447, 1448) entries
created by the summary processing modules 1440,
1444.

A reconciliation module 1424 verifies that the
daily input updates are accurately reflected in the

-179-

ledger postings. Furthermore, the reconciliation
module 1424 verifies that the GLI module 72 will
properly apply correct updates to a firm's general
ledger.

5   The reconciliation module 1424 is scheduled to run
       in batch at the end of the processing day, when the
       closing balance files of every interfacing
       transaction processing module have been sent. The
       transaction processing modules that reside on the
10   minicomputer will use a notification function to
       inform the GLI module when their balances are
       closed on any given day.

       The reconciliation module 1424 of the present
       invention comprises three basic reconciliation
15   attempts. First, the reconciliation module 1424
       determines whether the day's projected closing
       balances, are equal to the system's actual closing
       balances. Projected closing balances are
       calculated by adding the daily posting to the GLI
20   summary files 1445, 1446 to the previously stored
       opening balances. The summations are created on a
       general ledger account key basis and stored in a
       projected closing balances file 1442. An "actual"
       closing balances file 1430 contains data summed by
25   the reconciliation modual 1424 as originally sent
       from transaction processing modules.

       The projected and actual balances are separately
       summed on a general ledger account key basis. They
       are then compared to each other to detect
30   differences. For the differences discovered by the
       reconciliation module 1424, a reconciliation
       exception report 1432 is printed containing the
       general ledger account keys involved in the break,

the respective balances, their differences, and the
processing date. The breaks can be researched at
the trade level, by viewing the contents of the
detail file 1410. The reconciled, reclassified
5 closing balances are written to a reclassified
closing balance 1438.

The reconciliation attempts a second reconciliation
between data in the projected closing balance file
1402 and data in a general ledger balance file
10 1434. This reconciliation is performed daily to
compare the source system balances to their general
ledger equivalents. This ensures that the general
ledger is receiving the GLI summary postings
properly and that no erroneous general ledger
15 manual entries have been made.

If any differences are discovered, they are
reported in an error report by the reporting
module. This report contains the general ledger
account key, the respective balances, the
20 differences between the balances, and the
processing date. This information is provided to
support research of the breaks.

The reconciliation module 1424 performs a third "as
of" reconciliation every month to ensure that the
25 general ledger system is properly reflecting all
"as of" activity. The reconciliation is performed
after the "as of" posting process is closed (as a
accounting matter) for the prior accounting
period's "as of" adjustments. The reconciliation
30 module 1424 creates a projected "as of" net change
file 1436 which combines the closing periods
balances on the last day of the previous month with
all prior period "as of" adjustments applied.

Those adjusted balances are compared, on an account
key basis, to the general ledger balances for the
last day of the previous month 1431. If any
differences are discovered during the comparison,
5 an "as of" reconciliation error report is printed
by the report module 1432, containing information
to help trace the break's origin. The general
ledger account key, the respective balances, the
differences, and the processing date are all
10 printed to the report.

E. Implementing the Transaction Processor Feature Modules

The transaction processor of the present invention
can be implemented using any programming language
15 that is supported by the operating system of the
environment in which the program module resides.
In the present invention, the Seer Technologies
Inc. rules-based language and the computer-aided
software engineering facility sold under the
20 trademark of "High Productivity System" or "HPS"
was employed in implementing the transaction
processing modules. For background information on
the HPS rules-based language and CASE facility, the
reader is referred to the co-pending U.S.
25 application serial number 444,060, filed November
30, 1989 entitled, "Computer-Aided Software
Engineering Facility," that is hereby expressly
incorporated by reference.

-182-

What is Claimed is:

1.   A computer system for processing transaction
     information, which comprises

     a.   a transaction information entry module
     including:

               i.   a first processor having an
               input to receive information
               relevant to a plurality of
               individual transactions and for
               generation of individual
               transaction records, one
               corresponding to each one of the
               plurality of individual
               transactions, and

               ii. a transaction record file
               coupled to said first processor for
               input and indexed storage of each
               one of the individual transaction
               records;

     b.   a transaction record processing module
     coupled to said transaction record file and
     arranged to controllably access eac· one of
     the individual transaction records from said
     transaction record file, said transaction
     record processing module including:

i.   a second processor for
correlating each one of the
accessed transaction records to
individual transaction account
information and cumulative
transaction accounting and
inventory information, and

ii. a preselected set of
individual account and cumulative
accounting and inventory files
coupled to said second processor
for input and indexed storage of
correlated information to update
and record the individual
transaction account and cumulative
transaction accounting and
inventory information as a function
of the individual transaction
records;

c.   a communication module coupled to each
of said transaction information entry module
and said transaction record processing module
to transmit notification communications
between said transaction information entry
module and said transaction record processing
module, the notification communications
including notification of storage of
individual transaction records in said

-184-

transaction record file; and

d. said transaction record processing
module being responsive to the notification
communications to access the individual
transaction records from said transaction
record file, asynchronously to operation of
said transaction information entry module,
such that entry of information relevant to
each of the plurality of individual
transactions and correlating processing and
update and recordation of individual
transaction account and cumulative
transaction accounting and inventory
information, as a function of the individual
transaction records, is processed in an on-
line operation.

2. The computer system of claim 1, further
comprising:

a. a preselected set of reference data
bases containing attribute information
relating to data elements input to said first
processor in connection with the plurality of
transactions;

b. said first processor being coupled to
said set of data bases for accessing the
attribute information during generation of

the individual transaction records so as to
include relevant attribute information in
each one of the transaction records.

3.   The computer system of claim 1, wherein said
      first processor and said second processor
      together comprise a central processing unit
      having a multi-tasking capability.

4.   The computer system of claim 1, further
      comprising a computer network including a
      plurality of personal computers and at least
      one main data processing facility coupled to
      one another and wherein:

      a.   said transaction information entry
      module includes said plurality of personal
      computers for input of the information
      relevant to the plurality of individual
      transactions; and

      b.   said first and second processors
      together comprise said at least one main data
      processing facility in a multi-tasking mode
      of operation.

5.   The computer system of claim 4, wherein said
      at least one main data processing facility
      includes a fault-tolerant mini-computer and a

mainframe computer cooperatively operating
with one another as said first processor and
said second processor.

6.   The computer system of claim 1, wherein said
      transaction record processing module
      comprises independently, asynchronously
      operating sub-modules, each operating to
      receive the notification communications,
      access said transaction record file and
      perform correlating processing of each one of
      the transaction records in respect of
      preselected portions of the individual
      transaction account and cumulative
      transaction accounting and inventory
      information, respectively, and to update
      preselected ones of said set of individual
      account and cumulative accounting and
      inventory files.

7.   The computer system of claim 6, wherein each
      one of said sub-modules accesses a
      preselected portion of each transaction
      record, the preselected portion being
      relevant to the respective portion of the
      individual account and cumulative transaction
      accounting and inventory information for
      which said sub-module performs correlating
      processing.

8.   A transaction processor for processing input
     data comprising information on a plurality of
     individual executed transactions, the
     transaction processor comprising:

     a.   a first central file containing general
     product and market data related to
     transactions;

     b.   a second central file containing
     executed transaction records;

     c.   a result file for storing cumulative
     indications of expected subsequent business
     events relating to the plurality of
     individual executed transactions;

     d.   a transaction input module coupled to
     said first central file and said second
     central file, to receive the input data,
     process the input data using data from said
     first central file relevant to the plurality
     of individual executed transactions, to
     create a plurality of individual processed
     executed transaction records, one
     corresponding to each of the plurality of
     individual executed transactions and each
     including general product and market data
     relevant to one of the plurality of
     individual executed transactions, and to
     write each of the processed executed

-188-

transaction records in said second central
file;

e.   a transaction processing module being
coupled to said second central file to access
each of the executed transaction records from
said second central file being coupled to
said result file to update the cumulative
indication of executed subsequent business
events as a result of the plurality of
individual executed transactions; and

f.   a communication module coupled to each
of said transaction input module and said
transaction processing module to transmit
notification communications between said
transaction input module and said transaction
processing module, the notification
communications including notification of
storage of individual transaction records in
said second central file and to invoke said
transaction processing module to access
executed transaction records from said second
central file and to update said result file.

9.   A module for storing data relating to executed
transactions, said module comprising:

a.   a first database containing general
product information and routines to perform

general transaction related calculations on
the data relating to a plurality of executed
transactions;

b.   a second database to store transaction
records;

c.   a transaction processing module to
receive the data relating to each of the
plurality of executed transactions, being
coupled to said first database to access and
use general product information and routines
to validate and embellish the data related to
each of the plurality of executed
transactions, for creating a plurality of
executed transaction records, one for each of
the plurality of the executed transaction
records, and being coupled to said second
database, to store each of the executed
transaction records at a respective
preselected location in said second database;
and

d.   a notification module to output message
primitives each comprising an identification
of the executed transaction records and the
preselected location in the second database
of the executed transaction records.

10. A module for processing transaction input

-190-

comprising data on executed transactions, received from
more than one transmitting locations, the transaction
input module comprising:

   a.   a database containing general product
   information and routines to perform general
   transaction related calculations on the input
   data;

   b.   a database to store processed
   transaction input;

   c.   a file to store processed transaction
   input;

   d.   a first input processing module to
   receive the transaction input data from a
   first selected one of the more than one
   transmitting locations, process the data, and
   store the transaction data in said database,
   processing comprising validating the data
   against the general product information,
   embellishing the input data with general
   product information and calculating
   transaction related values;

   e.   a second input processing module to
   receive transaction input data from a second
   one of the more than one transmitting
   locations, process the data, and store the
   transaction data in said file, processing

comprises validating the data against the
general product information; and

f.   a breaksheet processing module to match
processed transaction data in said file
against processed transaction data in said
database and to create a breaksheet report,
the breaksheet report comprising dated
instances where data in said file has no
match against data in the said database.

11.   A cash management system module for maintaining
all cash flow movement related to executed
transactions, said cash management system module
comprising:

a.   a transaction information entry module
including:

i.   a transaction processor having
an input to receive information
relevant to a plurality of
individual transactions and for
generation of individual
transaction records, one
corresponding to each one of the
plurality of individual
transactions, and

ii.   an executed transaction record

-192-

file coupled to said first
processor for input and indexed
storage of each one of the
individual transaction records,
each record containing data on an
executed transaction including cash
due on an executed transaction;

b.   a payable and receivable file for
storing payable and receivable data created
to represent cash due on executed
transactions;

c.   a cash activity file containing records
detailing events related to cash flow
movement;

d.   a cash totals file containing current
cash balances in at least one cash account;

e.   a payable and receivable creation module
arranged to access records from said executed
transaction record file, process the accessed
records to create data representative of the
cash due on each of the executed transactions
and store data in said payable and receivable
file;

f.   a payable and receivable update module
arranged to accept data input on cash flow

movement and process that data to update the
records in said payable and receivable file,
update the cash balances in said cash totals
file and create a record of the cash movement
in said cash activity file, so that all cash
due relating to executed transactions is
cross referenced to cash flow movement;

g.   a communication module coupled to said
transaction entry module and said payable and
receivable creation module to transmit
notification communications between said
transaction entry module and said payable and
receivable creation module, the notification
communications including notification of
storage of individual transaction records in
said executed transaction record file;

h.   said payable and receivable creation
module being responsive to the notification
communications to access each of the
individual transaction records from said
transaction record file, asynchronously to
operation of said transaction information
entry module, such that entry of information
relevant to each of the plurality of
individual transactions and correlating
processing and update and recordation of
individual transaction account and cumulative
transaction accounting and inventory
information, as a function of the individual

transaction records, is processed in an on-
line operation.

12. The cash management module of claim 11, further
comprising a module to access said payable and
receivable file, said executed transaction file, said
cash totals file, and said cash activity file to create
reports on current bank account totals and projected
cash movement and totals.

13. A module arranged to track product holdings for an
organization, holdings comprising products purchased in
a transaction on a transaction execution date and to be
paid for on a transaction settlement date, said module
comprising:

> a.   a plurality of files to store data on
> the holdings as of transaction execution date
> and transaction settlement date, the files by
> transaction execution date comprising an
> account file indicating product ownership and
> a location file indicating the locations
> where account file products are actually
> located, the files by transaction settlement
> date comprising an account file indicating
> product ownership, and a file for the
> locations where the account file products are
> actually located;

> b.   a module to accept transaction data

input concerning executed transactions and
product movements, said module arranged to
create entries to the transaction execution
date and settlement date account and location
files to track the execution of transactions
and movement of transaction products; and

c.   a module arranged to read each record in
said transaction execution date location file
and said settlement date location file, match
each location entry against a counterpart
entry in the transaction execution date or
transaction settlement date account files,
create a dated record break entry in the
corresponding file when a counterpart entry
is not found and create a report, the report
comprising entries on each account file that
has no counterpart entry in said location
file and where the corresponding dated record
break was made.

14. A module arranged to track the liquidation of
product holdings for an organization having customer
accounts, holdings comprise products purchased in a
transaction on a transaction execution date that are to
be paid for and delivered on a transaction settlement
date, said module comprising:

a.   a plurality of lot files to store data
on product account holdings, each lot file

-196-

being arranged in a lot liquidation list, the
lot liquidation list comprising records on
account holdings ordered according to a
preselected strategy for liquidation by sale;

b.   said lot files arranged to created
liquidation lots as of transaction execution
date and as of transaction settlement date,
the files by transaction execution date
comprising a file for account lot liquidation
by list, a file for holdings that have been
liquidated from the lot liquidation lists,
and a file for cancelled, corrected and late
additions to the lot lists, the files by
transaction settlement date comprising a file
account lot liquidation by list, a file for
holdings that have been liquidated from the
lot liquidation lists, and a file for
cancelled, corrected and late additions to
the lot lists;

c.   a module to receive input, input
comprising executed trade data and product
movement data, and process that data to
create updates to the lot liquidation files
by transaction execution date or settlement
data according to the lot liquidation
strategy;

d.   a plurality of files to store data on
the holdings as of transaction execution date

-197-

and as of transaction settlement date, the
files as of transaction execution date
comprising a file for account holdings; the
files as of transaction settlement date
comprising a file for account holdings;

e.   a module arranged to read each record in
the transaction date account file and create
a corresponding record in the settlement date
file, and create lot liquidation entries for
the settlement date lot liquidation files;
and

f.   a module arranged to read each record in
the transaction date lot liquidation files
and the settlement date lot liquidation
files, match each location entry against a
counterpart entry in the transaction
execution date or transaction settlement date
account files, create a dated record break
entry in the corresponding file where the
match was expected and create a report,
report comprising entries on each file record
that does not match and where the
corresponding dated record break was made.

15. A clearance module arranged to process input,
input comprising data on executed transactions, the
clearance module comprising:

-198-

a.   a transaction information entry module
including:

      i.   a transaction processor having
      an input to receive information
      relevant to a plurality of
      individual transactions and for
      generation of individual
      transaction records, one
      corresponding to each one of the
      plurality of individual
      transactions, and

      ii.   an executed transaction record
      file coupled to said first
      processor for input and indexed
      storage of each one of the
      individual transaction records,
      said executed transaction record
      file containing a first record on
      each of a plurality of executed
      transactions, each transaction
      including a purchase and sale of a
      product;

b.   a clearance main file;

c.   a first clearance module arranged to
access each first record from said executed
transaction record file, and process each
first record to create a second record

denoting the expectation of the receipt and
delivery of the product relating to the
respective one of the plurality of executed
transactions, and writing each second record
to said clearance main file;

d.   a clearance line-up file;

e.   a second module arranged to access said
clearance main file to select certain ones of
the second records depicting executed
transactions that are expected to have
receipt and delay of a respective product
during a certain business day and write the
selected second records to said clearance
line up file;

f.   a third module arranged to receive input
concerning the settlement of the plurality of
executed transactions, settlement input
comprising data relevant to whether the
product of the executed trade was received or
delivered;

g.   a clearance detail file;

h.   said third module arranged to process
said input concerning the settlement of
executed transactions, processing comprised
of updating each of the plurality of records

-200-

in said clearance main file corresponding
with each data input, storing each data input
in said clearance detail file;

i.   a communication module coupled to said
transaction entry module and said first
clearance module to transmit notification
communications between said transaction entry
module and said first clearance module, the
notification communications including
notification of storage of individual
transaction records in said executed
transaction record file; and

j.   said first clearance module being
responsive to the notification communications
to access each of the individual transaction
records from said transaction record file,
asynchronously to operation of said
transaction information entry module, such
that entry of information relevant to each of
the plurality of individual transactions and
correlating processing and update and
recordation of individual transaction account
and cumulative transaction accounting and
inventory information, as a function of the
individual transaction records, is processed
in an on-line operation.

16.   The clearance module of claim 15 wherein said
second module is coupled to and arranged to output to a

CPU-linked bank processing module.

17. The clearance module of claim 15 further comprising:

    a. a pair up file; and

    b. a third module arranged to access said clearance main file and pair related transactions that are expected to settle, creating a single paired record in said the clearance main file and writing the components of the pair into said pair up file.

18. The clearance module of claim 15 further comprising:

    a. a failed file; and

    b. a fourth module to access said clearance main file and said line up file and extract transactions that failed to settle on their given settlement date, creating a failed transaction record for each fail and writing each to said fail file.

19. A entitlement module arranged to track the receipt

-202-

and payment of entitlements related to products bought
and sold in plurality a transactions relating to a
plurality of individual transaction accounts,
entitlements comprising dividends, interest and
redeemables related to the products, said module
comprising:

a.   a database of general product
entitlement information;

b.   an entitlement update module coupled to
the general product entitlement information
database, arranged to receive information on
entitlements and update said general product
entitlement database based upon the
information on entitlements;

c.   a database of executed transaction
information including information on products
bought and sold and respective individual
transaction accounts;

d.   an entitlement announcement file;

e.   an entitlement announcement module
arranged to create entitlement schedule lists
using the data in said general product
entitlement database and write the
entitlement schedule lists to said
entitlement announcement file;

f.   a proofsheet file;

g.   a proofsheet creation module arranged to
access said product entitlement schedule file
and said database of executed trades and to
create proofsheet records indicating
entitlements due on products bought and sold
in respect of individual transactions
accounts and being coupled to said proofsheet
file to write the created proofsheet records
to said proofsheet file;

h.   a receipt and reconciliation module
arranged to accept input on entitlements paid
and to process the input on entitlements paid
by linking the input on entitlements paid to
corresponding records in said proofsheet
file;

i.   said module arranged to create a list of
entitlements due to the individual
transactions accounts reports, claim letter;
and

j.   a distribution module to distribute the
entitlements paid to the individual
transaction accounts in accordance with the
list of entitlements due.

20. The entitlement module of claim 19, further

-204-

comprising an adjustment module being coupled to said
proofsheet file and to said entitlement list file
arranged to accept input data concerning alterations to
said proofsheet file and said entitlement list file and
alter or recreate said entitlement of proofsheet file
according to a preselected scheme.

21.   The entitlement module of claim 19, further
comprising:

    a.   an entitlement table;

    b.   a one-day table;

    c.   a distribution module being coupled to
    the proofsheet file arranged to read the
    contents of said proofsheet file, calculate
    entitlement due and distribute those
    entitlement in phases to a plurality of
    accounts, depending on type of account and
    type of product and being coupled to said
    entitlement history table write the records
    of entitlement due to said entitlement
    history table;

    d.   said distribution module being coupled
    to a one day table, write all entitlement due
    on predetermine days to said one-day table.

22.   The entitlement module of claim 19, and claim 21
further comprising:

a.   a due bill file; and

b.   a module arranged to access said
entitlement history table and create due bill
records for each entitlement due on trades
that have failed to clear on said settlement
date, and being coupled to said due bill file
output the records to said due bill file and
also output said created records formatted as
a due bill.

23.   A general ledger interface module comprising:

a.   a transaction information entry module
including:

     i.   a transaction processor having
     an input to receive information
     relevant to a plurality of
     individual transactions and for
     generation of individual
     transaction records, one
     corresponding to each one of the
     plurality of individual
     transactions, and

     ii.   an executed transaction record
     file coupled to said first
     processor for input and indexed

-206-

storage of each one of the
individual transaction records,
said executed transaction record
file containing a first record on
each of a plurality of executed
transactions, each transaction
including a purchase and sale of a
product;

b.   a source module coupled to said executed
transaction record file, arranged to output
said records;

c.   a reference file containing a list of
entries marking ledger journal entries to be
made for each of the plurality of the
preselected data inputs;

d.   an intermediate file;

e.   a journal processing module, coupled to
said source module, arranged to process said
source module output by accessing said
reference file and creating a record for each
data input listing the ledger journal entries
to be made from each data input and an
offsetting record for each data input listing
preselected offsetting ledger journal process
entries to be from each data input, said
journal processing module being coupled to an
intermediate file, said journal processing

module writing said records into said
intermediate file;

f.   a detail file;

g.   a detail explosion module coupled to
said intermediate file, designed to create
journal entry records for each record in said
intermediate file and, said detail explosion
module being coupled to said detail file, the
created records written by said detail
explosion module to said detail file;

h.   a summary journal file;

i.   a summary process module coupled to said
detail file arranged to sum the journal
entries according to a predetermined scheme
and, said summary process module being
coupled to said summary journal file, write
the summation records to said summary file;

j.   a plurality of general ledger files;

k.   general ledger module, coupled to said
summary journal file and connected to said
plurality of ledger journal files, arranged
to write the contents of the summary journal
file to said ledger journal file according to
a predetermined scheme;

-208-

l.   a communication module coupled to said
transaction entry module and said source
module to transmit notification
communications between said transaction entr··
module and said source module, the
notification communications including
notification of storage of individual
transaction records in said executed
transaction record file;

m.   said source module being responsive to
the notification communications to access
each of the individual transaction records
from said transaction record file,
asynchronously to operation of said
transaction information entry module, such
that entry of information relevant to each of
the plurality of individual transactions and
correlating processing and update and
recordation of individual transaction account
and cumulative transaction accounting and
inventory information, as a function of the
individual transaction records, is processed
in an on-line operation.

24. The general ledger interface module in claim 23
further comprising a control account module coupled to
said journal processing module and invoked by said
journal processing module to create an offsetting
record for each data input, listing the offsetting

ledger journal entries to be made from the data input
when the data input contains no data from which an
offsetting ledger journal entry can be created.

25.   The general ledger interface module in claim 23
further comprising:

    a.   an end of day closing balance file
    containing end of day closing transaction
    balance information;

    b.   a reconciliation module, coupled to said
    end of day closing balance file and coupled
    to said detail file, arranged to match
    records in said end of day closing balance
    file against records in said detail file,
    create reports on the results of the
    matching; and

    c.   a reconciliation break file;

    d.   said reconciliation module, being
    coupled to said reconciliation break file,
    arranged to write, in said reconciliation
    break file, all created records on non-
    matchings between records in said detail file
    and said end of day closing balance file.

26.   The general ledger interface module in claim 23

-210-

further comprising a reclassification module coupled to
a second input source arranged to reclassify entries in
said intermediate file based on classification updates.

27.   A notification and security system for a computer
network including a plurality of network objects,
network objects comprising computer terminals, users
and transaction processor modules, the notification and
security system comprising:

    a.   a database containing predetermined
    security and access information to establish
    links between a plurality of network objects
    on the network;

    b.   a plurality of interconnected routers,
    routers connected to transaction processor
    module objects and terminal objects;

    c.   a plurality of queues, each said queue
    comprising a memory, each queue coupled to a
    respective one of a router, a transaction
    processing module and a terminal object;

    d.   a profiler module arranged to establish
    and control communication links between said
    network objects, said profiler being coupled
    to said database, accessing data in said
    database to control the establishment of
    communication links between the network

objects and routers;

e.   said profiler being coupled to an input
queue to receive input messages from said
queue said messages comprising a source and
destination heading and a message command to
be interpreted by said profiler;

f.   said profiler arranged to process the
message and return a message to the source
object or other objects;

g.   said profiler arranged to process the
message and return the processed result in
the form of a message by writing the message
to said queue of the router to which the
profiler is coupled;

h.   said router arranged to read input
messages from a respective queue, check the
message's destination and write the message
to said respective queue of a respective
router coupled to a preselected one of said
network objects indicated on destination
heading of the message;

i.   said profiler arranged to process login
and logout requests sent from network objects
by matching the request message information
against preselected access data in said
database;

j.   said profiler arranged to add objects to
the network by processing said login requests
and adding successfully logged objects to the
database, and sending messages to other
network objects of a new object login;

k.   said transaction processing network
objects arranged to send communication
messages between themselves by combining the
data with a message header, indicating the
destination object and writing the message to
a respective queue of a respective router
connected to the source object.

28.   The communication and notification system of claim
27, further comprising:

a.   a plurality of PC interface modules in a
mini computer environment coupled to said
routers;

b.   a plurality of PC router modules in a PC
workstation environment, coupled to a PC
interface module, arranged to invoke the PC
interface module and transmit input from the
PC workstation environment to the
minicomputer environment; and

c.   a preselected set of reference data

bases containing attribute information
relating to data elements input to said first
processor in connection with the plurality of
transactions.

29. A store and forward module arranged to transmit
data between computer hardware environments, the store
and forward module comprising:

a. a first computer hardware environment
having a first environment file format
structure and at least one transaction
processing module operating to create
records;

b. a second computer hardware environment
having a second environment file format
structure;

c. a store and forward log, resident in the
first computer hardware environment,
containing individually numbered records on
executed transactions, each record containing
a code indicating a record number and at
least one transaction processing source
module in said first hardware environment
that is the source of the records;

d. a first environment transmission module,
resident in the first computer hardware
environment, being coupled to said store and

-214-

forward log, accessing said store and forward
log records and initiating transmission of
the records to the second computer hardware
environment;

e.   a store and forward module resident in
the second computer hardware environment,
coupled to said first environment processing
module, said store and forward module
receiving the store and forward log records
transmitted by said first environment
transmission module;

f.   a key file database coupled to said
store and forward module, containing pre-
selected translation and filing information
relevant to records transmitted by said first
environment transmission module and the
transaction processor source module on said
first hardware environment;

g.   said store and forward module arranged
to process said received records by the
record number of each record transmitted by
said first environment transaction module and
translating each transmitted record from said
first environment file format structure to
said second environment file format
structure, using key file translation
information;

h.   a plurality of second environment file
format data storage files;

i.   said store and forward processing module
being coupled to said second environment file
format data storage files, arranged to write
the translated records to said second
environment file format data storage files,
using filing information in said key files;

j.   a notification module, coupled to said
store and forward processing module, arranged
to generate sent messages containing the
second environment file format data storage
file locations of the new store and forward
log updates.

30.   A method of processing transaction data
comprising:

a.   a central database containing static
transaction related information; and

b.   transaction processing modules that
processes transaction data by receiving
transaction related data and processing that
transaction data by accessing the static
transaction related information in the
central database to embellish the transaction
data.

-216-

31. A method of transaction processing comprising:

a.   a transaction information entry module
including:

   i.   a transaction processing
   module having an input to receive
   information relevant to a plurality
   of individual transactions and for
   generation of individual
   transaction records, one
   corresponding to each one of the
   plurality of individual
   transactions, and

   ii. a transaction record file
   coupled to said first processor for
   input and indexed storage of each
   one of the individual transaction
   records;

b.   a database containing product attribute
information;

c.   a classification tree comprised of a
plurality of linked nodes, the nodes
containing information from said product
attribute information database, the
classification tree ordering the nodes
according to a predetermined classification

scheme based on product attributes;

d.   a traversal module arranged to read add,
delete or modify nodes in the classification
tree;

e.   said transaction processing module
processing said transaction input by invoking
said traversal module to traverse said
product classification tree and search the
tree on a preselected search criteria; and

f.   said transaction processing module
arranged to accept the results of said
product classification tree search and access
said database based on entries corresponding
to nodes in said product classification tree.

32.   A method of processing transaction input
comprising:

a.   a database containing product
information;

b.   a classification tree comprised of a
plurality of linked nodes, the nodes
containing information from the product
information, the classification tree ordering
the nodes according to a predetermined
classification scheme, the nodes coupled to

-218-

entries in the database;

c.   a traversal module arranged to read add,
delete or modify nodes in the classification
tree; and

d.   a data processing module coupled to the
product classification tree and the traversal
module and arranged to access the said
database by invoking the traversal module to
traverse the classification tree, read each
node and then access the corresponding
database entries.

33.   The computer system of claim 6 wherein:

a.   said transaction information entry
module receives information relevant to a
securities transaction including a purchase
and sale of a financial security and related
exchange of cash, on a trade date and
information on a future settlement date for
delivery of the financial security and
exchange of the cash, and wherein:

b.   said sub-modules include a cash
management sub-module and a financial
security clearance sub-module;

c.   said cash management sub-module includes

an integrated payable and receivable file and
performs transaction processing to record
cash due and cash payable and related
settlement date information from each one of
the transaction records;

d.   said security clearance module
comprises:

    i.   a clearance main file,

    ii.   a first module arranged to access
    each transaction record from said
    transaction record file and to process
    each transaction record to create a
    record denoting the settlement date of
    the receipt or delivery of the financial
    security of the respective transaction
    record and write each created record to
    said clearance main file,

    iii. a clearance line up file,

    iv.   a second module arranged to access
    the clearance main file, select created
    records indicating receipt or delivery
    of a financial security on a certain
    date and write the selected created
    records to said clearance line up file,

    v.   a second module arranged to receive

-220-

input concerning the settlement of
executed transactions, settlement input
comprising data relevant to whether the
product of the executed trade was
received or delivered,

vi.   a clearance detail file,

vii. said second module arranged to
process the input concerning the
settlement of executed transactions,
processing comprised of updating the
record in said clearance main file with
the data input, storing each data input
in said clearance detail file.

Fig 1

FIG 3

Fig. 4

FIG 4a

FIG 4b

Fig. 4 d

Fig. 4 e

FIG 5

(IO MANAGER)

FIG 6

150 — Transaction Processing Module (e.g. Trade entry Module)

154 — I/O Manager

via e.g. Stratus-Brand Link

Store and forward log file — 13

168 — Mini computer Resident Store and Forward Communication Module

Mini Computer environment

Store and forward Priority Sorter (ASYNC) — 160

Priority log files

162 164 166

170

152

172

174 — ASCII file

Key files

180

Mainframe environment

176 — Store and forward Controler Module

Translater module Translater module Translater module Translater module Translater module Translater module

178

188

EBCIDIC file — 174

UPDATER MODULE

NOTIFIER MODULE — 46

190

192

Mainframe resident Transaction Processing Module

44 — Massive Storage Tables

FIG 7

Country ≠ United States 250

N/A 296 Country = US 252

Type = Equity 254 280

Type ≠ Equity 256

Type = Equity 258 282

Type ≠ Equity 260

Issuer = Government 262

Issuer ≠ Government 264 284

Issuer = Government 266

Issuer ≠ Government 268 286

Product Issuer = Municip 270 288

Product ≠ Munic 272 290

Product ≠ Municip 274 292

Product ≠ Munic 276 294

FIG 8.

FIG 9

Fig 10

Ticket

PC Input Module

Error Checking

PC Router

Generic Distribution Mechanism

38

queue

TRADE ENTRY System

(continued)

FIG 11
page 1

Trade Entry
WAKE UP
module

352

Validation
• get customer/product information
• check fields/keys
• Relational type checking

354

general
Reference
Databases

76

figuration

Trade
extract
builder
(Batch

Data field

DERIVATION

general
extract

reference
file

357

Trade Generation
and Trade
number assignment
module

write fully
figured trade(s)
to log file

log
file

366

return Message
to PC via
Generic distribution
mechanism TRADE
IS Booked

(Continued
FIG 11
2/3
page 2

write fully figured trade to log file

log file

366

Updater
(Asynchronous processing)
Takes log file and adds files to executed trade file

378

Io manager

store and forward log

executed trade files

50

batch indexing log

375

382

NOTIFIER

Indexer

376

Indexes to Executed trade file

374

FIG 11
3/3
page 3

362

356

general ledger for contracts

380

if Trade requires trade generation

no

384

Generate Trades

364

Book Trade

FIG 11a

FIG 1
page 1

414 — BOOK FLOOR TRADE

LOG FILE — 412

416 — UPDATER (ASYNCHRONOUS)

418 — I O MANAGER

43 — STORE AND FORWARD LOG

FLOOR ENTRY TRADE FILE — 420

FIG 12

Page 2

FLOOR
ENTRY
FILES
~420

DESK
ENTRY
FILES
~450

EXECUTED
TRADE
FILES
~50

IO MANAGER
MODULE
~468

RECONCILIATION
GATHER DATA
MODULE
~460

MATCH TRADE
MODULE
470

GENERAL
REFERENCE
DATABASES

REPORTING
MODULE
476

UNMATCHED
TRADES ON
ETF
~474

REPORT
~478

TRADE REQUEST INFO — 480

GDM — 38

484 — RETRIEVE TRADE INFO

ID MANAGER MODULE — 486

EXECUTED TRADE FILES

493 — DISPLAY TRADE BY TRADE TYPE: EQUITY, FUTURE, OPTION, DEBT MORTGAGED - BAILED SECURITIES

MAINFRAME RESIDENT TRADE TABLES — 490

SCREEN DISPLAY — 492

DO UNTIL EXIT — 506

TRADE MANAGE- MENT REVIEW — 494

CANCEL/ CORRECT HISTORY — 498

MULTIPLE TRADE SEARCH — 496

GIVE UP HISTORY — 500

"WHEN ISSUE" HISTORY — 502

ALL TRADES WITH SAME SEQUENCE NUMBER — 504

FIG. 11

FIG 16
a

TRADE DATE

SETTLEMENT DATE

602 CUSTOMER ACCOUNT TRADE DATE TABLE

608 CUSTOMER ACCOUNT SETTLEMENT DATE TABLE

614 PRODUCT ACCOUNT MEMO TABLE

CUSTOMER

FIRM

604 FIRM ACCOUNT TRADE DATE TABLE

610 FIRM ACCOUNT SETTLEMENT DATE TABLE

LOCATION

606 LOCATION TRADE DATE TABLE

612 LOCATION SETTLEMENT DATE TABLE

62

624 P&B BALANCE ACTIVITY MODULE

POSITION & BALANCE MANAGER MODULE

600

TRANSACTION ACTIVITY TABLE

620

618 TRADE REFERENCE TABLE

P&B TRADE PROCESSOR MODULE

616

ACTIVITY TABLE

622

64

TRADE ENTRY MODULES

48

CUSTOMER ACCOUNTS MODULE

68

FIRM INVENTORY MODULE

66

CLEARANCE MODULE

52

DIR MODULE

70

FIG 17

Firm Account Settlement Date Table					610
Company	Product	Account	Sub-Type		Quantity
Firm-NYO	USTB 1/18	TRD-ACCT-A			+5,000,000

610a

Location Settlement Date Table						612
Company	Product	Account	Sub-Type	Position Code	Quantity	
Firm-NYO	USTB 1/18	IRV		FREE	-2,000,000	
Firm-NYO	USTB 1/18	MHT		FREE	-3,000,000	

612a

612b

FIG 18a

Customer Account Settlement Date Table 608

Company	Product	Accnt	Sub-Type	Quantity	
Firm-NYO	EXXON	CUST-A	MARGIN	+1,500	608a
Firm-NYO	EXXON	CUST-A	CASH	+3,000	608b

Product Account Memo Table 614

Company	Product	Accnt	Sub-Type	Position Code	Quantity	ASSOC ACCNT #	
Firm-NYO	EXXON	CUST-A	MARGIN	SEG	+ 500	DTC	614a
Firm-NYO	EXXON	CUST-A	CASH	SEG	+3,000	DTC	614b
Firm-NYO	EXXON	CUST-A	MARGIN	FREE	+1,000		614c

Location Account Settlement Date Table 612

Company	Product	Accnt	Sub-Type	Position Code	Quantity	
Firm-NYO	EXXON	DTC		FREE	-1,000	612c
Firm-NYO	EXXON	DTC		SEG	-3,500	612d

FIG 18b

TRANS CODE	SETTLE DATE	B/S NATURE	ORIG QTY	ORIG PROC	DESCR	TRADE TYPE	ACCT	SUB	TYPE ACT
011	90/2/5	SELL/ AGENT	1500	450.00	IBM	CUST	A	MARGIN	SELL
022	90/2/5	BUY/ AGENT	4500	1350.00	IBM	BRKR	ML		BUY
011	90/2/5	SELL/ AGENT	3000	900.00	IBM	CUST	A	CASH	SELL

630
632
634

Customer Account Settlement Date Table 608

Company	Product	Accnt	Sub-Type	Quantity	
Firm-NYO	IBM	CUST-A	MARGIN	+1,500	608c
Firm-NYO	IBM	CUST-A	CASH	+3,000	608d

Product Account Memo Table 614

Company	Product	Accnt	Sub-Type	Position Code	Quantity	ASSOC ACCNT #
Firm-NYO	IBM	CUST-A	MARGIN	FREE	+1,500	614e
Firm-NYO	IBM	CUST-A	CASH	FREE	+3,000	614f

Location Account Settlement Date Table 612

Company	Product	Accnt	Sub-Type	Position Code	Quantity	
Firm-NYO	IBM	ML		F/R	-4,500	612e

FIG 18c

CLEARANCE MODULE DATA

TRANS CODE	SETTLE DATE	CLEAR NUMBER	CLEAR QTY	ORIG PROC	DESCR	TRADE TYPE	ACCT	CLEAR HOUSE	TYPE ACT
055	90/2/5	0161	4500	1350.00	IBM	BRKR	ML	DTC	REC

~636

CUSTOMER ACCOUNTS MODULE DATA

TRANS CODE	SETTLE DATE	QTY	DESCR	TRADE TYPE	ACCOUNT	SUB TYPE	TYPE ACTIVITY
0XX	90/2/5	500	IBM	CUST	CUST-A	MARGIN	SEG
0XX	90/2/5	3000	IBM	CUST	CUST-A	CASH	SEG

638

~640

Product Account Memo Table   614

Company	Product	Accnt	Sub-Type	Position Code	Quantity	ASSOC ACCNT #	
Firm-NYO	IBM	CUST-A	MARGIN	FREE	− 500		614g
Firm-NYO	IBM	CUST-A	MARGIN	SEG	+ 500	DTC	614h
Firm-NYO	IBM	CUST-A	CASH	FREE	−3,000		614i
Firm-NYO	IBM	CUST-A	CASH	SEG	+3,000	DTC	614j
Firm-NYO	IBM	CUST-A	MARGIN	FREE	+1,500		614e
Firm-NYO	IBM	CUST-A	CASH	FREE	+3,000		614f

Location Account Settlement Date Table   612

Company	Product	Accnt	Sub-Type	Position Code	Quantity	
Firm-NYO	IBM	DTC		FREE	+ 500	612k
Firm-NYO	IBM	DTC		SEG	− 500	612j
Firm-NYO	IBM	DTC		FREE	+3,000	612i
Firm-NYO	IBM	DTC		SEG	−3,000	612h
Firm-NYO	IBM	DTC		FREE	−4,500	612g
Firm-NYO	IBM	DTC		F/R	+4,500	612f
Firm-NYO	IBM	DTC		F/R	−4,500	612e

FIG 18d

FIG 19a
Page 1

710 — PEB SUB-MANAGER (MF) → (Continued on next page)

TRANS CODE & TRADE DATA

NUMBERED TRANSACTION ACTIVITY RECORDS

712 — FORMAT P&B ACTIVITY

NEXT TRANS NUMBER

REQUEST FOR NUMBER

TRANS CODE & TRADE DATA

TRANS & ACTIVITIES

724 — BUILD P&B TRANS NUMBER

714 — ACTIVITY AND POSITION CODE ACCESSOR RULE

ACTIVITY DATA

TRANS CODE

720 — ACCESS P&B ACTIVITY TABLE

622 — P&B ACTIVITY TABLE

600

64

FIG 19
PAGE 2

FIG 19a
PAGE 3

FIG 19
PAGE 1

CONTINUED

STORE & FORWARD ～ 41

46 ～ STORE & FORWARD NOTIFIER

64

710 ～ P&B SUB MANAGER

ERROR REPORT FILE ～ 740

～ 600

TRANSACTION ACTIVITY

RETURN CODE

726 ～ UPDATE P&B FILES

TRANSACTION ACTIVITY

ACCESS&UPDATE TRANS-ACTIVITY LOG FILE

RETURN CODE

P&B TRANS ACTIVITY LOG FILE ～ 620

734

TRANSACTION ACTIVITY

RETURN CODE

732 ～ ACCESS&UPDATE CUST T/D FILE

CUSTOMER TRADE DATE POSITION FILE ～ 612

732

ACCESS&UPDATE CUST S/D FILE

CUSTOMER SETTLE DATE POSITION FILE ～ 616

～ 62

732 ～ ACCESS&UPDATE LOCATION T/D FILE

LOCATION TRADE DATE POSITION FILE

～ 618

732

ACCESS&UPDATE LOCATION S/D FILE

LOCATION SETTLE DATE POSITION FILE ～ 620

610

732 ～ ACCESS&UPDATE FIRM T/D FILE

FIRM TRADE DATE POSITION FILE

732

ACCESS&UPDATE FIRM S/D FILE

FIRM SETTLE DATE POSITION FILE ～ 614

ACCESS&UPDATE MEMO FILE

MEMO SETTLE DATE POSITION FILE ～ 622

732

FIG 19
PAGE 2

FIG 20

	Trade date	product	quantity	Price	COST	Action	Account
836 T3	3/2	IBM	25	10.5C	$262.50	Sell	Account #1

834 T2	3/1	IBM	50	11.00	$55.00	Buy	Account #1
832 T1	3/1	IBM	100	10.00	1000.00	Buy	Account #1

828

829

826

838
840

T1
T2

FIFO OPEN LOT

T1

842

FIFO CLOSED LOT

CANCELED LOT

826a

828a

842a

838	T1	3/2	IBM	100 75	10.00	Buy	Acct #1	838a T1	3/2	IBM	100	10.00	-75	10.50	3/2
840	T2	3/1	IBM	50	11.00	Buy	Acct #1	840a							

FIG 20a

826

842 828

838 | T1 | 3/1 | IBM | 100 0 | BUY | 10.00 | } | { Account 1
840 | T2 | 3/1 | IBM | TO 100 | BUY | 10.10 | } | { ACCNT 1
Trade Date | Prod. | Quantity | Action | Price | | Account

OPEN LOT FILE FIFO

843 | T1 | 3/1 | IBM | 100-100 | 10.00 | 10.20 | T3 | 3/8
| T2 | 3/1 | IBM | 100-50 | 10.10 | 10.20 | T3 | 3/8
Tr. Date | Prod. | Quantity | Purchase Price | Sell Price | Liquidation Trans.

FIFO CLOSED LOT

"AS OF" record

| T4* | 2/28* | IBM | 100 | 9.50 | BUY | ACCOUNT # 1 |

Trade | Date | Product | Quantity | Price

826c

T4	2/28	IBM	100 0	BUY	9.50	Accnt # 1
T1	3/1	IBM	100 50	BUY	10.00	Accnt # 1
T2	3/1	IBM	100	BUY	10.10	Accnt # 1

FIFO OPEN LOT

828c

| T4 | 2/28 | IBM | 100-100 | 9.50 | 10.20 | T3 |
| T1 | 3/1 | IBM | 100-50 | 10.00 | 10.20 | T3 |

FIFO CLOSED LOT

892

832 | T1 | 3/1 | IBM | 100 | 10.00 | BUY | Accnt # 1
834 | T2 | 3/1 | IBM | 100 | 10.10 | BUY | Accnt # 1
836 | T3 | 3/3 | IBM | 150 | 10.20 | SELL | Accnt # 1
890 | T4 | AS OF 2/28 | IBM | 100 | 9.50 | BUY | Accnt # 1

Trade | Date | Product | Quantity | Price | Action | Account

FIG 20b

FIG 21a

MARGIN REQUIREMENTS (PCS) 962

MARGIN OVER-RIDE 963

974

EXCEPTION TABLE 970

NYSE Extention

NOTICE AUTHORIZATION TABLE 975

ACTIVITY MATRIX 960

TO GLI INTERFACE 932

M.A. Mod 973

INTEREST RATES TABLE 964

PRINTING 978

FIG 216

FIRM PRICE MANAGEMENT DATABASE 210

INITIAL MARGIN MODULE 942

MAINTENANCE MARGIN MODULE (HOUSE) 944

MARGIN SETTLEMENT BALANCES MODULE 946

Exception / TO NYSE EXTENTION 974

EXCEPTION PROCESSING MODULE 948

NOTICE AUTHORIZATION MODULE 950

DEBIT/CREDIT INTEREST MODULE 954

TO (CMS) Payment 939

TO CMS Payment 934

REPORTING MODULE 976

CUSTOMER ACTIVITY TABLE 952

932

P&B Module

FPM 210 964

60 Clearance Main

TO (GLI INT) 952 972

INTEREST DETAIL TABLE

MARGIN MARK TO MARKET 632

CD 602

SETTLE DATE 608

MEMO TABLE 614

77

P Master 206

FPM 210

208

TRADE DATE

CMS

CUSTOMER ACCOUNT MONEY BALANCES 965

CUSTOMER SUB ACCOUNT MONEY BALANCES 967

AMOUNT YEAR TO DATE BALANCE 966

CUSTOMER STATUS 969

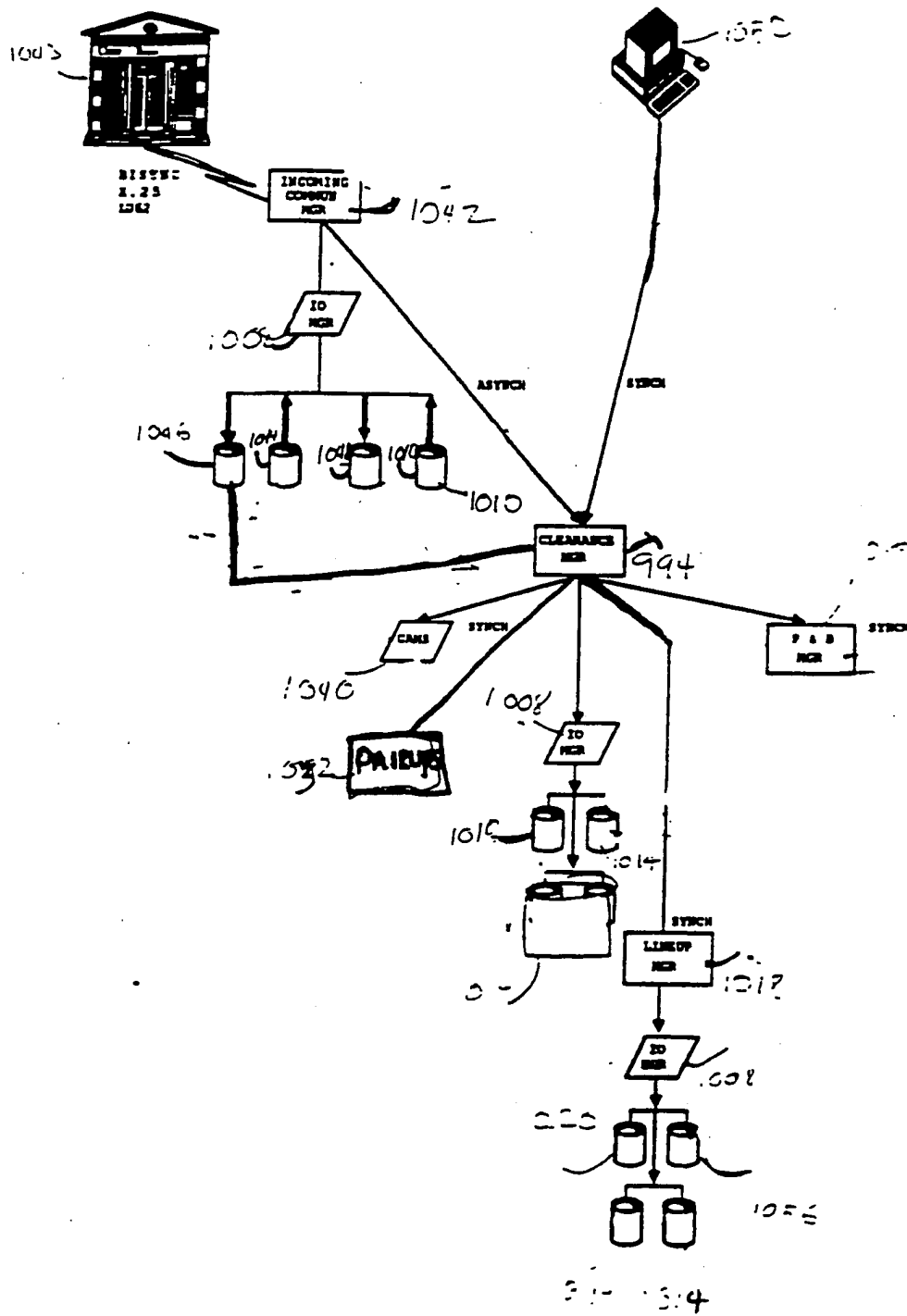SUB A YtoD BAL 968

NOTES 971

FIG 22a

Trade Gets Paired Off

FIG
22b

TRADE CLEARS

FIG 22C

FAILED TRADE CLEARS

FIG 22d

FIG 23b

604 610
62
1135
602 608
606
PEB TABLE 612 614

Proofsheet
Entitlement
Gathering
Module 1270

1066 1064
Clearing
Fail
Detail
Clearing
Fail
File

1276

1274 Intermediate
Log

Proofsheet
Matching
Module

1284 Proofsheet
Detail
Collection
Module

1290

Due Bills
Module

1137

1298

1297 Header
file

Due
Bill
File

Proofsheet Table 1282

Proofsheet
Details 1289

Calc.
D.B.

Determine
D.B
Type

Create Proofsheet Module

Proofsheet

1292 1294

1136

1296

Clearance
Module 52

FIG 23c

FIG 23d

Minicomputer
Entitlement
adjustments.
1300

Store and Forward
Module
42

34

36

Store and forward
main Control
41

STORE and Forward
NOTIFIER
Module
46

1306
adjustment
process
Storage table

1307

1310
Pre-
Processing
Module

Proofsheet
Adjustment
Driver Module
1308

1326
Criticalness
Pre processing
module

1338

P.C.S
Type
1328

1171
Batch
Processing
file

1134
Entitlement
Announcement
Processing

1312
Current
Proofsheet
Cycle adjustment
Module

1314
Past
Proofsheet
Cycle adjustment
Module

Change
EASy files

1135

1320

1324

1290
Due
Bills

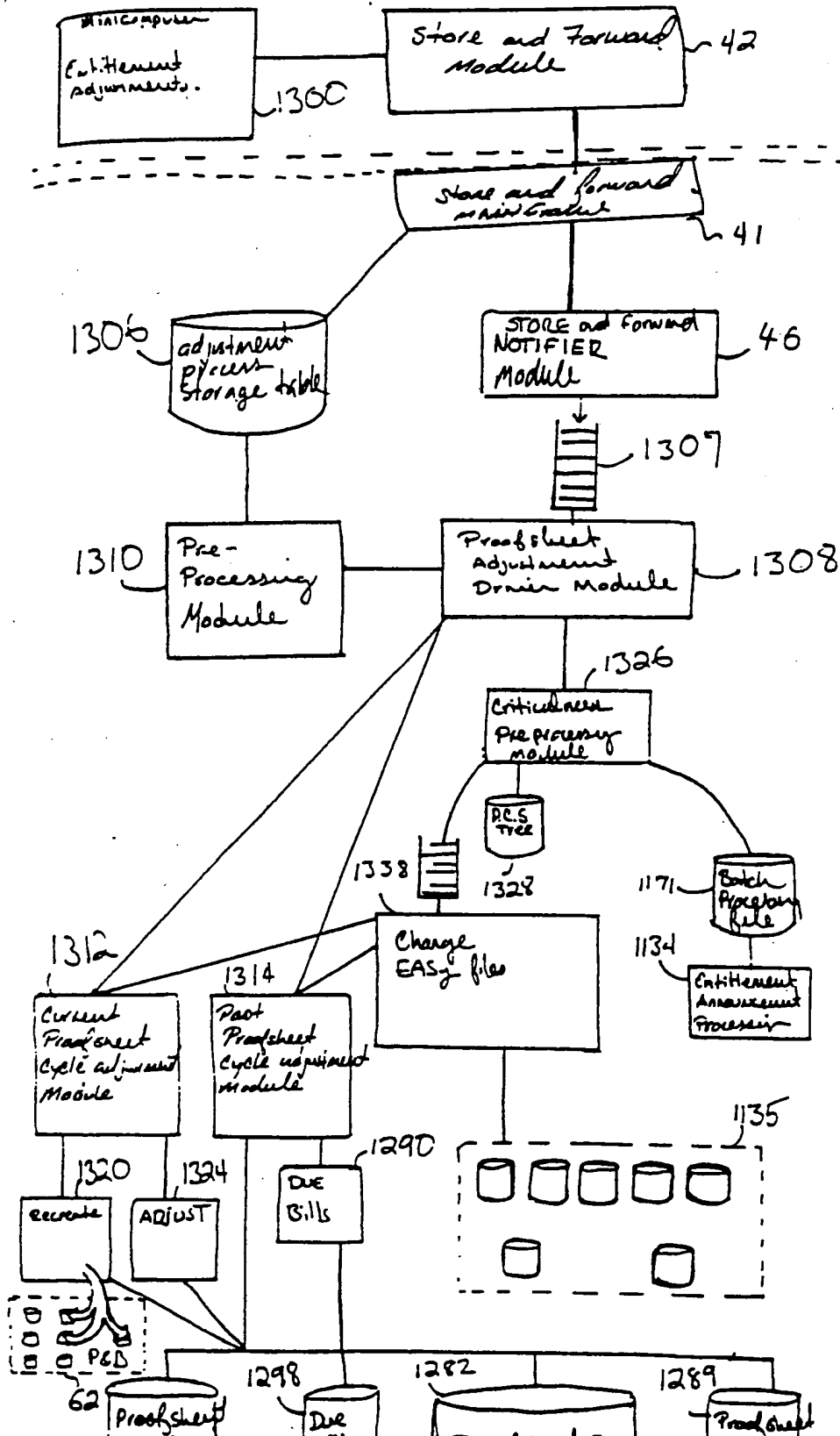Recreate

ADJUST

P&D
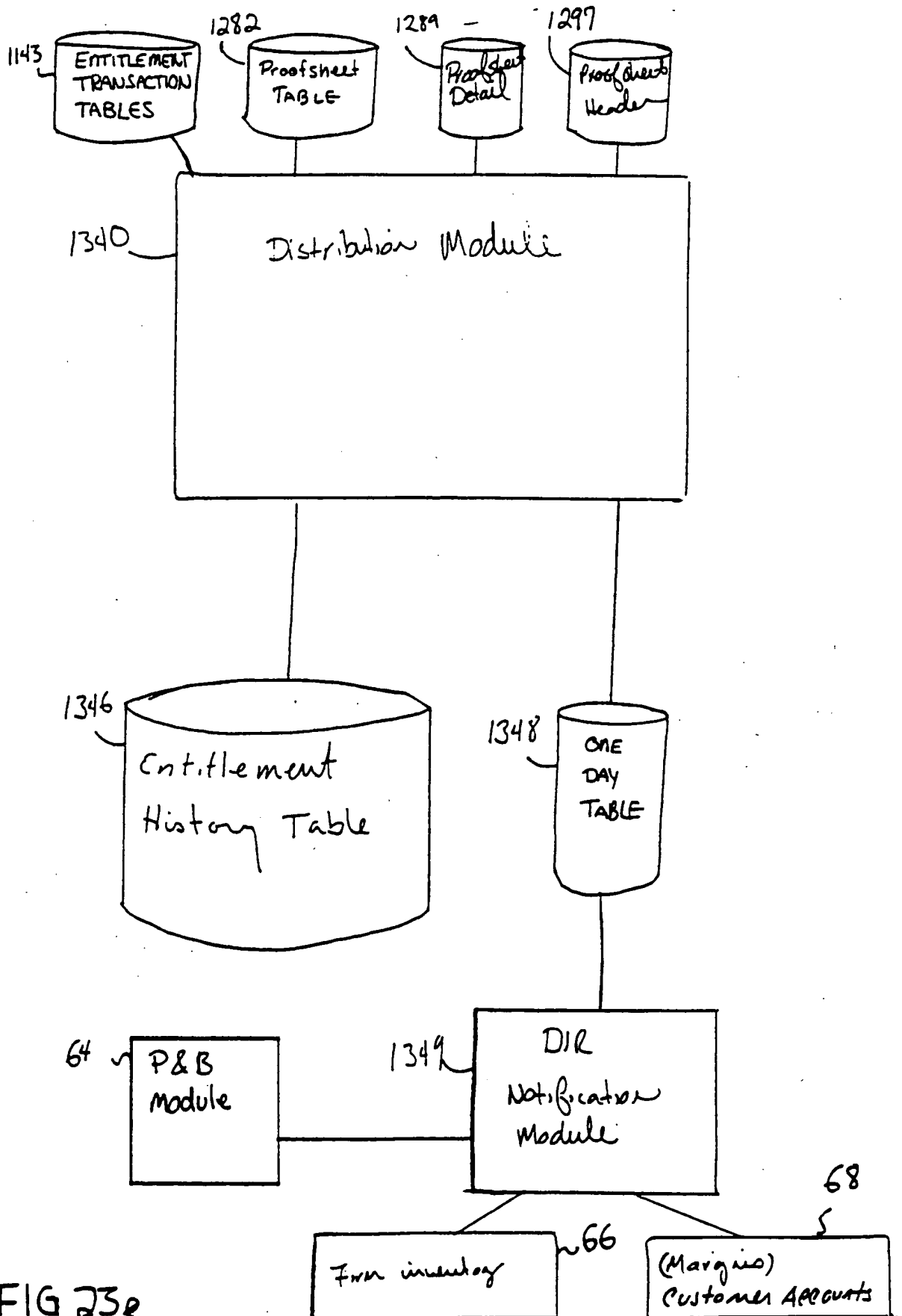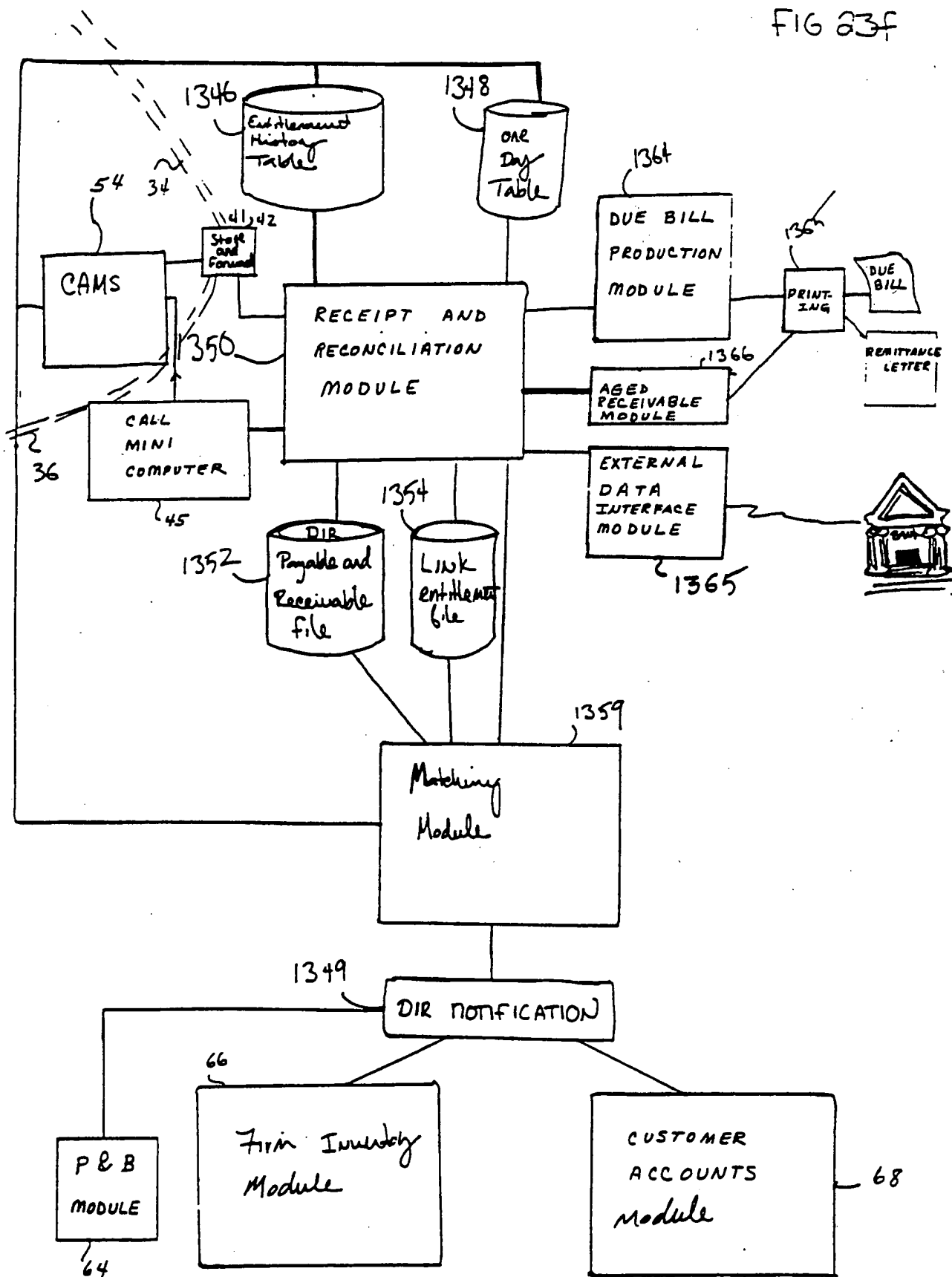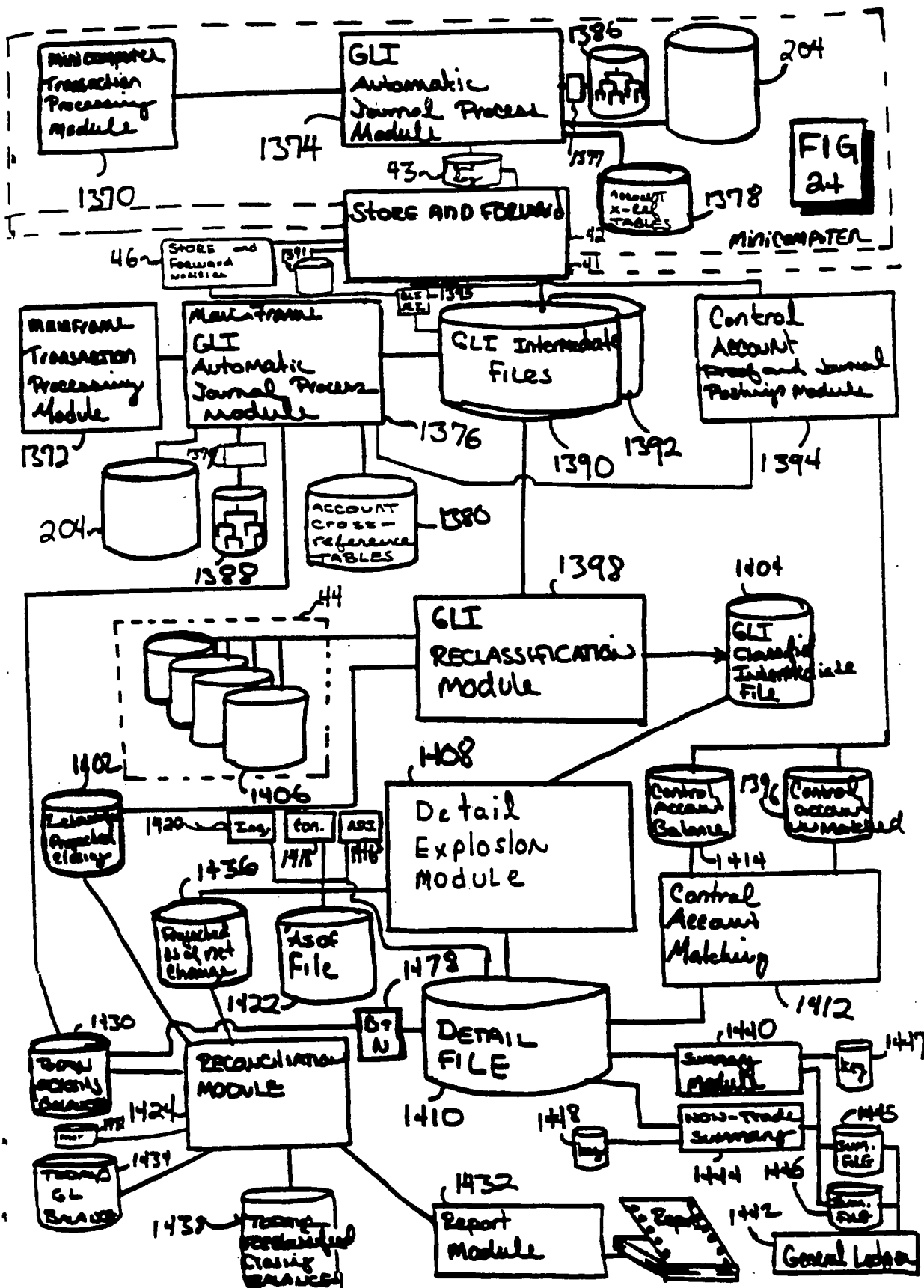
62 Proofsheet

1298 Due

1282

1289 Proofsheet

FIG 23e

FIG 23f

FIG 24

INTERNATIONAL SEARCH REPORT

International Application No. PCT/US91/06279

I. CLASSIFICATION OF SUBJECT MATTER (if several classification symbols apply, indicate all) [6]

According to International Patent Classification (IPC) or to both National Classification and IPC

IPC(5): G06F 15/21

II. FIELDS SEARCHED

Minimum Documentation Searched [7]

Classification System	Classification Symbols
U.S.Cl	364/200,900 (MS FILE)

Documentation Searched other than Minimum Documentation
to the Extent that such Documents are Included in the Fields Searched [8]

III. DOCUMENTS CONSIDERED TO BE RELEVANT [9]

Category [*]	Citation of Document, [11] with indication, where appropriate, of the relevant passages [12]	Relevant to Claim No. [13]
Y	US,A, 4,947,367 (CHANG) 07 AUGUST 1990, see entire document	29
Y	US,A, 4,903,201 (WAGNER) 20 FEBRUARY 1990 see entire document	1-7,9,15-18, 30 and 33
Y	US,A, 4,870,610 (BELFER) 26 SEPTEMBER 1989 see entire document	29
Y	US,A, 4,851,999 (MORIYAMA) 25 JULY 1989, see entire document	1-7,23-26, 32 and 33
Y	US,A, 4,815,030 (CROSS) 21 MARCH 1989, see entire document	1-7,10 and 33
Y	US,A, 4,646,229 (BOYLE) 24 FEBRUARY 1987, see entire document	8
Y	US,A, 4,631,664 (BACHMAN) 23 DECEMBER 1986, see entire document.	31

* Special categories of cited documents: [10]

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

IV. CERTIFICATION

Date of the Actual Completion of the International Search	Date of Mailing of this International Search Report
12 DECEMBER 1991	3 0 DEC 1991
International Searching Authority	Signature of Authorized Officer
ISA/US	MARIA N. Von BUHR -Examiner

FURTHER INFORMATION CONTINUED FROM THE SECOND SHEET	

Y	US,A, 4,623,964 (GETZ) 18 NOVEMBER 1986, see entire document	15-18,27 and 28
Y	US,A, 4,597,046 (MUSMANNO) 24 JUNE 1986, see entire document	8 and 15-18
Y	US,A, 4,554,418 (TOY) 19 NOVEMBER 1985, see entire document	1-7 and 33
Y	US,A, 4,479,196 (FERRER) 23 OCTOBER 1984, see entire document	31
Y	US,A, 4,412,287 (BRADDOCK) 25 OCTOBER 1983, see entire document.	13 and 14

V. ☐ OBSERVATIONS WHERE CERTAIN CLAIMS WERE FOUND UNSEARCHABLE [1]

This international search report has not been established in respect of certain claims under Article 17(2) (a) for the following reasons:

1. ☐ Claim numbers   , because they relate to subject matter [12] not required to be searched by this Authority, namely:

2. ☐ Claim numbers .... , because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out [13], specifically:

3. ☐ Claim numbers_____, because they are dependent claims not drafted in accordance with the second and third sentences of PCT Rule 6.4(a).

VI. ☐ OBSERVATIONS WHERE UNITY OF INVENTION IS LACKING [2]

This International Searching Authority found multiple inventions in this international application as follows:

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims of the international application.

2. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims of the international application for which fees were paid, specifically claims:

3. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claim numbers:

4. ☐ As all searchable claims could be searched without effort justifying an additional fee, the International Searching Authority did not invite payment of any additional fee.

Remark on Protest

☐ The additional search fees were accompanied by applicant's protest.

☐ No protest accompanied the payment of additional search fees.

III. DOCUMENTS CONSIDERED TO BE RELEVANT (CONTINUED FROM THE SECOND SHEET)		
Category*	Citation of Document, with indication, where appropriate, of the relevant passages	Relevant to Claim No
Y	US,A, 4,376,978 (MUSMANNO) 15 MARCH 1983, see entire document	11 and 12
Y	US,A, 4,346,442 (MUSMANNO) 24 AUGUST 1982, see entire document.	11 and 12
A	US,A, 4,153,931 (GREEN) 08 MAY 1979, see entire document	1-33
Y	US,A, 4,135,241 (STANIS) 16 JANUARY 1979, see abstract; column 10; columns 46-74	32
Y	US,A, 3,596,256 (ALPERT) 27 JULY 1971, see entire document	27 and 28
Y	US,A, ,3,573,747 (ADAMS) 06 APRIL 1971, see entire document.	13 and 14
Y	US,A, 3,297,992 (McDONALD) 10 JANUARY 1967 see columns 99-110.	19-26